KOFAX

Kofax SignDoc®
E-sign with Kofax

Kofax SignDoc 1.3.1

Developer's Guide

© 2017 Kofax. All rights reserved.

Kofax is a trademark of Kofax Inc., registered in the U.S. and/or other countries. All other trademarks are the property of their respective owners. No part of this publication may be reproduced, stored, or transmitted in any form without the prior written permission of Kofax.

Kofax SignDoc 1.3.1 - Developer's Guide
Saved 19-Apr-2017

Table of Contents

I Kofax SignDoc REST Interface 9

1 REST URL ... 9
2 REST Error Response ... 9
3 Error Code Information ... 10
4 Authentication Token ... 10
5 Authentication via API Key ... 12
6 Representation Formats ... 12
7 Swagger UI ... 12
8 User Roles and Permissions ... 13
9 Overview of the Differences Between v4 and v5 ... 14

II REST API Reference v5 15

1 Account Requests ... 15
- Create an Account ... 15
  - Parameters ... 16
  - Fields ... 17
  - Response ... 17
- Update an Account ... 18
  - Parameters ... 19
  - Response ... 19
- Delete an Account ... 19
  - Parameters ... 20
  - Response ... 20
- Get a Single Account ... 20
  - Parameters ... 21
  - Response ... 22
- Import License to Account ... 22
  - Parameters ... 23
  - Response ... 23
- Get all Accounts ... 23
  - Response ... 24
- Get Number of Accounts ... 24
  - Response ... 25

2 Configuration Requests ... 25
- Delete Configuration Settings ... 25
- Get Configuration Settings ... 26
- Set Configuration Settings ... 28
- Get Configuration Setting Descriptions ... 29

3 Document Requests ... 31
- Add Document to Package ... 31
  - Parameters ... 32
  - Response ... 33
  - Fields ... 33
- Update a Document ... 34
  - Parameters ... 35

 © 2017 Kofax

Response ... 35
Fields ... 36
Delete a Document ... 37
Parameters ... 37
Response ... 38
Get a Single Document ... 38
Parameters ... 38
Response ... 39
4 Event Requests ... 40
Trigger an Event ... 41
Parameters ... 43
Response ... 43
5 Field Requests ... 43
Add Checkbox to Document ... 44
Parameters ... 45
Fields ... 45
Get a Single Checkbox ... 46
Parameters ... 47
Response ... 47
Update a Checkbox with Form Data ... 47
Parameters ... 48
Fields ... 49
Response ... 50
Get All Fields ... 50
Parameters ... 51
Response ... 51
Delete a Field ... 51
Parameters ... 52
Response ... 52
Add Signature Field to Document ... 52
Parameters ... 53
Response ... 54
Fields ... 54
Get a Single Signature Field ... 55
Parameters ... 56
Response ... 56
Update a Signature Field with Form Data ... 57
Parameters ... 57
Fields ... 58
Response ... 59
Add Text Field to Document ... 59
Parameters ... 60
Response ... 60
Fields ... 61
Update a Text Field with Form Data ... 62
Parameters ... 62
Fields ... 63
Get a Single Text Field ... 64
Parameters ... 65
Response ... 65
6 Signing Package Requests ... 66
Create a New Signing Package ... 66
Parameters ... 67
Response ... 67
Fields ... 68
Example Response Body ... 69

Example Body ... 69
Create Signing Package Detailed Scenarios ... 69
Parameters Overview ... 70
Query Parameters ... 70
Body Parameters ... 70
Parameters in More Detail ... 70
Parameter Use Cases ... 71
Signing Package Properties Overview ... 71
Create Signing Packages with Word ... 80
Word Config ... 80
Principle Process ... 81
Scenarios ... 82
Scenario 1 - Create Signing Package from Scratch ... 82
Scenario 2 - Create Signing Package from Template ... 83
Scenario 3 - Create Signing Package from Template with Metadata ... 83
Scenario 4 - Create Signing Package with Word Document ... 85
Complete Signing Package Structure ... 86
Delete a Signing Package ... 88
Parameters ... 89
Update a Signing Package ... 89
Parameters ... 90
Response ... 90
Fields ... 90
Update an Existing Signature Field with Form Data ... 91
Parameters ... 92
Response ... 93
Get a Single Signing Package ... 93
Parameters ... 94
Response ... 95
Get Audit Trail ... 97
Parameters ... 98
Response ... 98
Get a List of Packages ... 100
Parameters ... 100
Response ... 100
Schedule a Signing Package ... 101
Parameters ... 102
Prepare Signing Session ... 102
Parameters ... 103
Response ... 104
7 Reminder Requests ... 105
Add Reminder to Package ... 105
Parameters ... 106
Response ... 106
Fields ... 107
Update a Reminder ... 107
Parameters ... 108
Response ... 108
Fields ... 109
Delete a Reminder ... 109
Parameters ... 110
Response ... 110
8 Signer Requests ... 110
Add Signer to Package ... 110
Parameters ... 111
Response ... 112

 © 2017 Kofax

Fields ... 112
Get a Single Signer ... 113
Parameter ... 114
Response ... 114
Delete a Signer ... 115
Parameters ... 116
Response ... 116
Update a Signer ... 116
Parameters ... 117
Response ... 117
Fields ... 118
Create a Signing Authentication Token ... 119
Parameters ... 120
Response ... 120
9 System Requests ... 120
Get Time Zones ... 120
Response ... 121
Get the Version of Kofax SignDoc Distribution ... 121
Response ... 122
10 Team Requests ... 122
Add Team to Account ... 122
Parameters ... 123
Fields ... 123
Response ... 124
Update a Team ... 124
Parameters ... 125
Fields ... 125
Response ... 125
Delete Team from Account ... 125
Parameters ... 126
Response ... 126
Remove User from Team ... 126
Parameters ... 127
Response ... 127
Add User to Team ... 128
Parameters ... 128
Response ... 129
11 User Requests ... 129
Create an Authentication Token ... 129
Parameters ... 130
Response ... 130
Get Number of Users ... 131
Parameters ... 131
Response ... 132
Create 'password recovery' Notification for a User ... 132
Parameters ... 132
Response ... 133
Refresh a new Authentication Token ... 133
Response ... 133
Get Current Server Administrator ... 134
Response ... 135
Create Server Administrator ... 135
Parameters ... 136
Fields ... 136
Response ... 137
Get all Server Administrators ... 137

Response ... 138
Delete a Server Administrator ... 138
Parameters ... 138
Response ... 139
Get a Single Server Administrator ... 139
Parameters ... 140
Response ... 140
Update a Server Administrator ... 140
Parameters ... 141
Fields ... 141
Response ... 142
Invite or Reinvite a Server Administrator ... 142
Parameters ... 143
Response ... 143
Reset Password of a Server Administrator ... 143
Parameters ... 144
Response ... 144
Add User to Account ... 144
Parameters ... 145
Fields ... 145
Response ... 146
Get Current User ... 146
Response ... 147
Get all Users ... 147
Parameters ... 148
Response ... 148
Get a Single User ... 149
Parameters ... 149
Response ... 150
Update a User ... 150
Parameters ... 151
Fields ... 151
Response ... 152
Delete a User ... 152
Parameters ... 153
Response ... 153
Invite or Reinvite a User ... 153
Parameters ... 154
Response ... 154
Reset Password ... 154
Parameters ... 155
Response ... 155
Add Role to User ... 156
Parameters ... 156
Response ... 157
Remove Role from User ... 157
Parameters ... 157
Response ... 158

III Kofax SignDoc Plugins 158

1 Plugin Development ... 158

Plugin Interface ... 158
Implementation ... 159
How SignDoc Uses Plugins ... 159

2 Plugin Packaging ... 160

3 Deployment 161
4 Configuration 161
Loading the Plugin 161
Enabling a Plugin 162
Configuring a Plugin 162
5 Available Plugins 163
Notification Plugin 163
Trusted Service Provider 164
IV Appendix 168
1 Contact Information 168
2 Trademarks 168
3 Copyright Notice 168
Index 169

1 Kofax SignDoc REST Interface

The Kofax SignDoc REST Interface provides a simplified possibility to work with Signing Packages, Accounts and Teams from different clients via the HTTP(S) protocol. The information exchanged between client and application server is typically in JSON format, but could be also XML.

This programming guide provides information on how to work with the Kofax SignDoc REST API. It covers the user roles defined by Kofax SignDoc and how they are reflected on the REST API as well as how to easily integrate Kofax SignDoc in your own Application. There also is a complete REST API description on each individual REST request in the chapters that follow. The Kofax SignDoc REST API can be consumed by any programming language able to use HTTP so. It requires nothing more than an API valid authentication to enable a user to interact with Kofax SignDoc in a headless manner.

1.1 REST URL

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

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

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

whereas path is divided in the parts

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

The context is cirrus by default, the rest part is fixed, followed by the version-number v5. The resource part identifies the requested resource, e.g. package.

Example

```
https://localhost:8080/cirrus/rest/v5/package
```

1.2 REST Error Response

HTTP is based on the exchange of representations, and that applies to errors as well. When a server encounters an error, either because of problems with the request that a client submitted or because of problems within the server, always return a representation that reflects the state of the error condition. This includes the response status code, response headers, and a body containing the description of the error. For errors due to client inputs, return a representation with a 4xx status code. For errors due to server implementation or its current state, return a representation with a 5xx status code.

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

Type	Elements	Value Type	Default Value	Description
RestMsgList	list	RestMsg	n/a	List of RestMsg elements

 © 2017 Kofax

Type	Elements	Value Type	Default Value	Description
RestMsg	code	number	n/a	Message code
	message	string	n/a	Description
	type	Message.TYPE	n/a	Can be ERROR, WARNING or INFO

Example Response Body (JSON)

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

1.3 Error Code Information

The Kofax SignDoc REST API's 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.

- 400 (Bad Request) means that 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) – This indicates that the authentication or authorization required for the API call did not pass.
- 403 (Forbidden) – This indicates that the authentication token or password is expired.
- 404 (NotFound) – This indicates that the resource being accessed does not exist. This might occur on GET or DELETE requests.
- 405 (MethodNotAllowed) – This generally indicates that the HTTP Method (GET, PUT, POST or DELETE) is not valid for this resource URI.
- 415 (UnsupportedMediaType) – This indicates that the data type of some data in the request is not supported.
- 500 (InternalServerError) – This indicates that an internal server-side error has occurred.
- 503 (Service Unavailable) – This indicates that the server is currently busy to process multiple requests at the same time. If a status code 503 occurs try again in a few seconds.

1.4 Authentication Token

For each request on the Kofax SignDoc REST API the user can authenticate via the Create an Authentication Token request (/rest/v5/users/authentication). This request takes username 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:

ew0KICAiYWNjb3VudElEIiA6ICJhY2NvdW50MyIsDQogICJhY2NvdW50TmFtZSIgOiAiYWNjaXVudDMiLA0KICAidXNlcklkIiA6ICJ1c2VyMjQiLA0KICAidXNlck5hbWUiIDogInVzZXIxMzQiLA0KICAiZU1haWwiIDogInVzZXIyM0Brc2QuY29tIiwNCiAgInJvbGVzIiA6IFsgIlVTRVIiLCAiVEVBTU1HUiIsICJBRE1JTiIgXSwNCiAgImV4cCIgOiAxNDQ3MDg5NzQ5NDA1LA0KICAiaWF0IiA6IDE0NDcwNzUzNDk0MDUNCn0=.64HScbedftEqYvMWXyiLT9a/oajzv97wQTbrvOS97e4=

The part highlighted in blue 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, ensuring that no changes can be made on the token. The hash is highlighted in orange.

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 Account Administrator rights can only use his token with the exact same permissions.

In the same manner the signer can authenticate via Create a Signing Authentication Token (see chapter Create a Signing Authentication Token). Signer's Authentication Token will be returned as X-S-Auth-Token response header.

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

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

NOTE

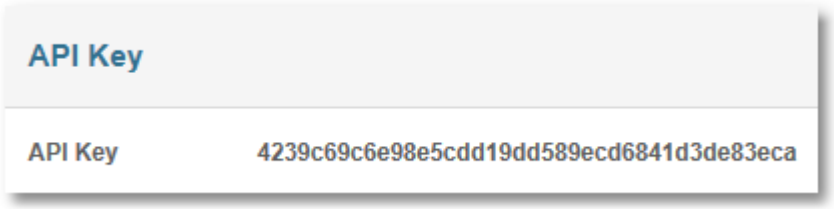
If a request to a Kofax SignDoc REST API contains more than one authentication token, Kofax SignDoc Standard Server takes the first one according to the following order:

1. X-S-Auth-Token
2. X-Auth-Token
3. API-key

 © 2017 Kofax

1.5 Authentication via API Key

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



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

1.6 Representation Formats

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

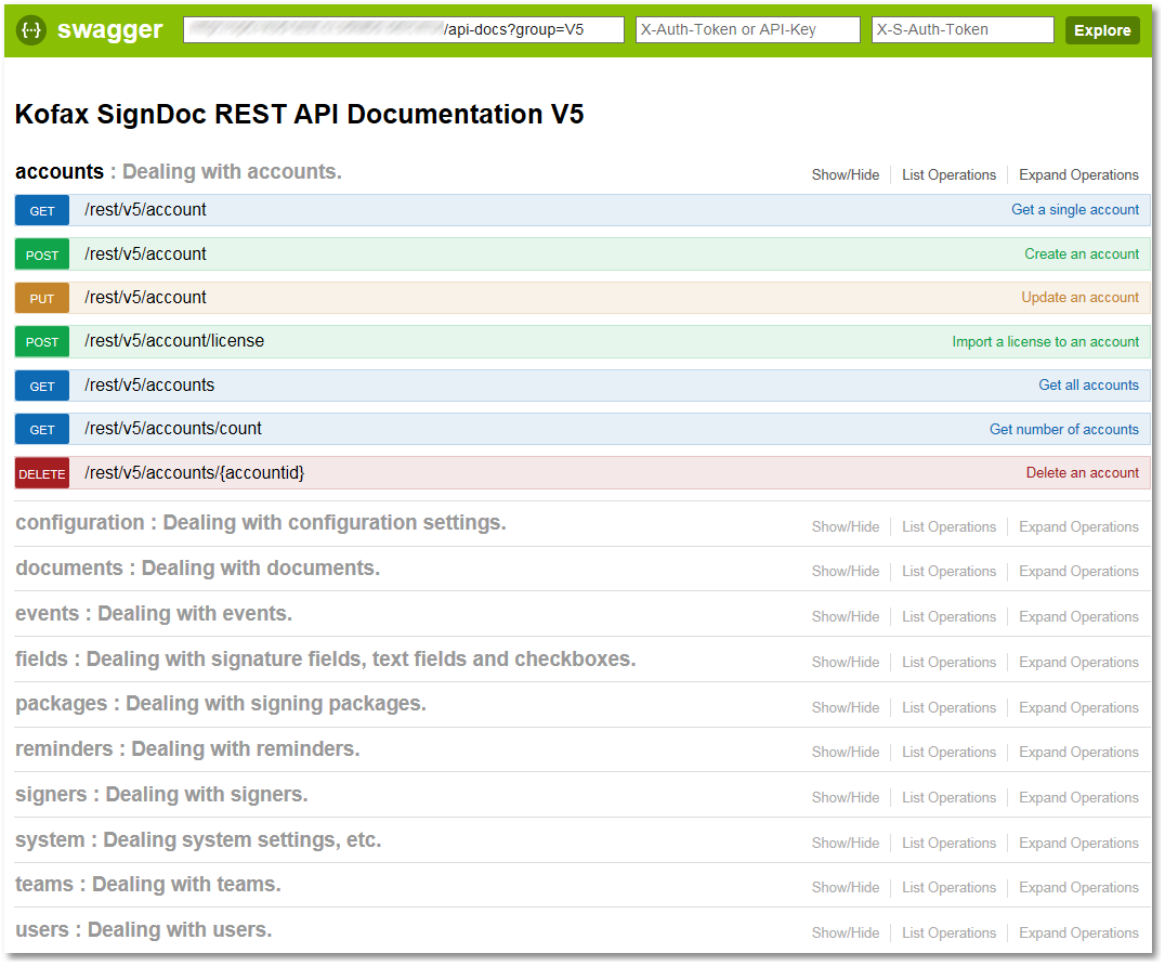
- by an extension postfixed to the URL (".json" or ".xml"), if using a simple web browser without the option to manipulate the Accept-Header suffix the URL with an extension ".json" or ".xml" to receive the desired format
- by the accept header found in the HTTP request
- by default as JSON

1.7 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 integrated into the server code. To experiment with the Kofax SignDoc 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 it parameters.



1.8 User Roles and Permissions

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

The following roles are currently defined for Kofax SignDoc:

Role	Description	Authorized Tasks
Super User	Super User is the first account present on a Kofax SignDoc distribution to enable a customer to create accounts.	With the role Super User you can: • Create accounts • Update accounts • Delete accounts • Query accounts
Account Administrator	Will initiate the Kofax SignDoc sequence to create and edit Signing	Same privileges as User, Team Member and Team Manager.

 © 2017 Kofax

Role	Description	Authorized Tasks
	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
User	Will initialize the Kofax SignDoc 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
Team Manager	Assigned Team Manager for a team. Administrates a team.	The Team Manager: • Invites Users to become a Team Member

1.9 Overview of the Differences Between v4 and v5

The following table describes the differences between REST API version 4 and version 5.

v4	v5
Update a signature field with form data `POST /rest/v4/packages/{packageid}/documents/{documentid}/signaturefields/{fieldid}`	Update a signature field (with RestSignatureFieldInput) `PUT /rest/v5/packages/{packageid}/documents/{documentid}/signaturefields/{fieldid}`
Update a text field with form data `POST /rest/v4/packages/{packageid}/documents/{documentid}/textfields/{fieldid}`	Update a text field (with RestTextFieldInput) `PUT /rest/v5/packages/{packageid}/documents/{documentid}/textfields/{fieldid}`
Get audit trail returns RestAuditTrail `GET /rest/v4/packages/{packageid}/audittrail`	Get audit trail returns RestAuditTrailOutput `GET /rest/v4/packages/{packageid}/audittrail`
`n/a`	Prepare signing session `POST /rest/v5/packages/{packageid}/signingsession/{signtype}`
`n/a`	Internal call in order to keep sdweb session alive

v4	v5
	`GET /rest/v5/system/keepalivesdw` `keepAliveSDW`

2 REST API Reference v5

Account Requests
Configuration Requests
Document Requests
Event Requests
Field Requests
Signing Package Requests
Reminder Requests
Signer Requests
System Requests
Team Requests
User Requests

2.1 Account Requests

Create an Account
Update an Account
Delete an Account
Get a Single Account
Import License to Account
Get all Accounts
Get Number of Accounts

2.1.1 Create an Account

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

 © 2017 Kofax

NOTE

For this request a valid authentication and Super User rights are necessary.

URL:	**/rest/v5/account**
Producible Formats:	JSON, XML
Return Formats:	JSON, XML
Request Header(s):	Accept: application/json, application/xml
HTTP Method:	POST

Example Request URL

```
POST https://localhost:8080/cirrus/rest/v5/account
```

Example Headers

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

Example Request Body (JSON)

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

2.1.1.1 Parameters

Body Parameters

Key	Required	Value Type	Default Value	Description
account	yes	RestAccountInput	n/a	Represents a RestAccountInput object either in JSON or XML

2.1.1.2 Fields

Mandatory Fields

Key	Value Type	Description
name	string	Name of account
license	string	Base64 encoded license

Available Fields

Key	Value Type	Description
id	string	ID of the account
name	string	Name of the account
company	string	Company of the account
state	ERestAccountState	State of the account
notificationsEnabled	boolean	Whether notifications are send or not
esignConsent	string	E-sign consent shown right before the signing ceremony
dnsLabel	string	The DNS label of account
contactInformation	string	Contact information of account
publicKey	string	Public key of account
signingCertificate	string	Contact information of account
users	RestUserInput	A list of all users of the account
teams	RestTeamInput	A list of all teams of the account
timeZone	TimeZone	Time zone of the account
license	string	Base64 encoded license for the account

2.1.1.3 Response

The response status is 201 if the account was successfully created, otherwise a Kofax SignDoc Status Code is returned together with the explaining message(s).

 © 2017 Kofax

2.1.2 Update an Account

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

NOTE

For this request a valid authentication and Account Administrator rights are necessary.

URL:	**/rest/v5/account**
Producible Formats:	JSON, XML
Return Formats:	JSON, XML
Request Header(s):	Accept: application/json, application/xml
HTTP Method:	PUT

Example Request URL

```
PUT https://localhost:8080/cirrus/rest/v5/account
```

Example Headers

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

Example Request Body (JSON)

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

2.1.2.1 Parameters

Query Parameters

Key	Required	Value Type	Default Value	Description
accountid	yes	string	n/a	The account ID. This parameter can be omitted if the user is not bound to an account. This is the case for users with the role SUPERUSER.

Body Parameters

Key	Required	Value Type	Default Value	Description
newAccount	yes	RestAccountInput	n/a	Input JSON or XML string of account

2.1.2.2 Response

The response status is 200 if the account was successfully updated, otherwise a Kofax SignDoc Status Code is returned together with the explaining message(s).

2.1.3 Delete an Account

This request deletes the existing account specified by ID.

NOTE

For this request a valid authentication and Super User rights are necessary.

URL:	**/rest/v5/accounts/{accountid}**
Return Formats:	JSON, XML
Request Header(s):	Accept: application/json, application/xml
HTTP Method:	DELETE

Example Request URL

```
DELETE https://localhost:8080/cirrus/rest/v5/accounts/account1
```

 © 2017 Kofax

Example Headers

```
Accept: application/json
```

2.1.3.1 Parameters

Path Parameters

Key	Required	Value Type	Default Value	Description
accountid	yes	string	n/a	The account ID. This parameter can be omitted if the user is not bound to an account. This is the case for users with the role SUPERUSER.

2.1.3.2 Response

The response status is 200 if the account was successfully deleted, otherwise a Kofax SignDoc Status Code is returned together with the explaining message(s).

2.1.4 Get a Single Account

Returns a single account specified by a given account ID. By default a flat account object with only essential information is returned. The accountFilter parameter allows to customize the response body of the request. Depending on a comma separated list of filter options the response body can return additional information. The ALL filter option allows a user to query the account object in its entirety.

NOTE

For this request a valid authentication and Super User rights are necessary.

URL:	**/rest/v5/account**
Producible Formats:	JSON, XML
Return Formats:	JSON, XML
Request Header(s):	Accept: application/json, application/xml
HTTP Method:	GET

Example Request URL

```
GET https://localhost:8080/cirrus/rest/v5/account
```

Example Headers

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

Example Response Body (JSON)


```
{
  "id": "account",
  "name": "account",
  "company": "account",
  "state": "ACTIVE",
  "notificationsEnabled": true,
  "esignConsent": "IMPORTANT! This is an example e-sign consent text. Replace this text
with your own legal text in your account preferences according to your local laws: This e-
sign consent (\"Consent\") applies to all Communications for services and accounts offered
or accessible through Kofax SignDoc that are not otherwise governed by the terms and
conditions of an electronic disclosure and consent. By selecting the "I agree"
acknowledgment on this page, you agree to sign documents electronically and understand
that the provision of such signatures constitutes a legally binding transaction according
to the applicable local e-sign legislature.",
  "users": [
    {
      "id": "user1",
      "name": "user1",
      "email": "user1@ksd.com",
      "state": "ACTIVE",
      "url": "/rest/v5/users/user1",
      "roles": [
        "USER",
        "TEAMMGR",
        "ADMIN"
      ],
  "teams": [
    {
      "id": "team1",
      "name": "Team_1 of Tenant_1 2015/10/21 11:47:29",
      "url": "/rest/v5/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": "/rest/v5/account?accountid=account3",
  "timeZone": {
    "timeZoneId": "Europe/Berlin",
    "displayName": "(GMT +01:00) Amsterdam, Berlin, Bern, Rome, Stockholm, Vienna"
  }
}
```


2.1.4.1 Parameters

Query Parameters

Key	Required	Value Type	Default Value	Description
accountid	no	string	n/a	The account ID. This parameter can be omitted if the user is not bound to an account. This is the case for users with the role SUPERUSER.
accountFilter	no	string	n/a	Available account filter options: TEAMS, USERS, NONE
useIntId	no	boolean	false	The provided accountId is the internal id which was returned by get audit trail method.

2.1.4.2 Response

The response status is 200 if the account was queried successfully, otherwise a Kofax SignDoc Status Code is returned together with the explaining message(s).

2.1.5 Import License to Account

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

NOTE

For this request a valid authentication and Account Administrator or Super User rights are necessary.

URL:	**/rest/v5/account/license**
Producible Formats:	JSON, XML
Return Formats:	JSON, XML
Request Header(s):	Accept: application/json, application/xml
HTTP Method:	POST

Example Request URL

```
POST https://localhost:8080/cirrus/rest/v5/account/license
```

Example Headers

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

2.1.5.1 Parameters

Query Parameters

Key	Required	Value Type	Default Value	Description
accountid	no	string	n/a	The account ID. This parameter can be omitted if the user is not bound to an account. This is the case for users with the role SUPERUSER.
license	yes	string	n/a	The Base64 encoded license for the account

2.1.5.2 Response

The response status is 201 if the license was successfully imported, otherwise a Kofax SignDoc Status Code is returned together with the explaining message(s).

2.1.6 Get all Accounts

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

NOTE

For this request a valid session and Super User rights are necessary.

URL:	**/rest/v5/accounts**
Producible Formats:	JSON, XML
Return Formats:	JSON, XML
Request Header(s):	Accept: application/json, application/xml
HTTP Method:	GET

Example Request URL

```
GET https://localhost:8080/cirrus/rest/v5/accounts
```

 © 2017 Kofax

Example Headers

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

Example Response Body (JSON)


```
[
  {
    "id": "account1",
    "name": " account1",
    "company": " account1",
    "state": "ACTIVE",
    "licenseInfo": {
      "expiryDate": 1451550974321,
      "licensedUsers": 10,
      "currentUsers": 8,
      "licensedPackages": 100,
      "processedPackages": 5,
      "licensedDocuments": 100,
      "processedDocuments": 5,
      "accountInformation": "some tenant specific information",
      "state": "VALID"
    },
    "url": "/rest/v5/account?accountid= account1"
  },
  {
    "id": " account2",
    "name": " account2",
    "state": "ACTIVE",
    "licenseInfo": {
      "expiryDate": 1451550975192,
      "licensedUsers": 100,
      "currentUsers": 1,
      "processedPackages": 0,
      "processedDocuments": 0,
      "accountInformation": "Enter customer number",
      "state": "VALID"
    },
    "url": "/rest/v5/account?accountid= account2"
  }
]
```


2.1.6.1 Response

The response status is 200 if the accounts were queried successfully, otherwise a Kofax SignDoc Status Code is returned together with the explaining message(s).

2.1.7 Get Number of Accounts

Returns the number of accounts.

NOTE

For this request a valid authentication and Super User rights are necessary.

URL:	**/rest/v5/accounts/count**
Producible Formats:	JSON, XML

Return Formats:	JSON, XML
Request Header(s):	Accept: application/json, application/xml
HTTP Method:	GET

Example Request URL

```
GET https://localhost:8080/cirrus/rest/v5/accounts/count
```

Example Headers

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

Example Response Body (JSON)

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

2.1.7.1 Response

The response status is 200 if the RestCount was queried successfully, otherwise a Kofax SignDoc Status Code is returned together with the explaining message(s).

2.2 Configuration Requests

Delete Configuration Settings
Get Configuration Settings
Set Configuration Settings
Get Configuration Setting Descriptions

2.2.1 Delete Configuration Settings

Delete one or more configuration values.

```
DELETE
   http(s)://domainOrIPAddress:port/context/rest/v5/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.

Example (JSON input)

```
POST .../rest/v5/configuration
```

with Body:

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

The requesting user user must have the permission to edit the configuration settings. The request is rejected if at least one 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.

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."
    }
  ]
}
```

2.2.2 Get Configuration Settings

Global System settings and also **Account specific settings** can be retrieved as Server Administrator, as Account Administrator, as User or as Signing Session (depending from the necessary authorization in the appropriate setting description).

```
GET http(s)://domainOrIPAddress:port/context/rest/v5/configuration/
   [?][accountid={accountid}][&][startswith={prefix}][&]
   [endswith={suffix}][&][cache={true|false}][&]effective={true|
   false}]
```

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, e.g. 'plugins'. In this case all configuration settings (which the caller is allowed to read) whose key start with 'plugins' are returned.

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

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

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

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

The **effective** flag indicates 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 then the global (account independent) value and last the default value if it is specified in the configuration description.

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

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

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

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

Example (JSON output)

```
GET .../rest/v5/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

 © 2017 Kofax

Reason	Error Code
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"
    }
  ]
}
```

2.2.3 Set Configuration Settings

Set one or more configuration values.

```
POST http(s)://domainOrIPAddress:port/context/rest/v5/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".

Example (JSON input**)**

```
POST .../rest/v5/configuration
```

with Body:

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

The requesting user user must have the permission to change the configuration settings. The request is rejected if at least one 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

Reason	Error Code
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."
    }
  ]
}
```

2.2.4 Get Configuration Setting Descriptions

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

```
GET
   http(s)://domainOrIPAddress:port/context/rest/v5/configuration/descriptions[?][accountid={accountid}][&][startswith={prefix}][&][endswith={suffix}][&][locale={locale}]
```

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.

 © 2017 Kofax

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

Example (JSON output)

```
GET .../v5/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."
    }
  ]
}
```

2.3 Document Requests

Add Document to Package
Update a Document
Delete a Document
Get a Single Document

2.3.1 Add Document to Package

Adds a new document to an existing signing package. An input JSON or XML string specifying the details of the document has to be provided as a request parameter. The specification has to provide at least one document content including the content attribute (the document bytes encoded as base64) of the document and a document name and a filename. The order is by default the current document size plus one.

NOTE

For this request a valid authentication and User rights are necessary.

URL:	**/rest/v5/packages/ {packageid}/document**
Producible Formats:	JSON, XML
Return Formats:	JSON, XML
Request Header(s):	Accept: application/json, application/xml

 © 2017 Kofax

HTTP Method:	POST

Example Request URL

```
POST https://localhost:8080/cirrus/rest/v5/packages/1001/document
```

Example Headers

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

Example Request Body (JSON)

```
{
  "name": "PDF example document",
  "fileName": "document.pdf",
  "format": "PDF",
  "content": " base64 encoded PDF",
  "signatureFields": [
  {
    "id": "signature-1",
    "page": 1,
    "height": 100,
    "width": 100,
    "posx": 100,
    "posy": 300,
    "required": "true",
    "label": "Signature field 1",
    "signingModeOptions": [
      "PH",
      "C2S"
    ]
  }
]
}
```

2.3.1.1 Parameters

Path Parameters

Key	Required	Value Type	Default Value	Description
packageid	yes	string	n/a	ID of the signing package the user wants to add a reminder

Query Parameters

Key	Required	Value Type	Default Value	Description
autoprep	no	boolean	false	Whether a document should be auto prepared or not. An auto prepared document will have a signature field automatically added on each page.

Body Parameters

Key	Required	Value Type	Default Value	Description
document	yes	RestDocument	null	Represents a RestDocument object either in JSON or XML.

2.3.1.2 Response

The response status is 201 (CREATED) if the document was created, otherwise a Kofax SignDoc Status Code is returned together with the explaining message(s).

2.3.1.3 Fields

Mandatory Fields

Key	Value Type	Description
content	string	Actual document in Base64 format

Available Fields

Key	Value Type	Description
id	string	The id of the document. id must be unique (within one package) and it must consist of characters from A-Z a-z 0-9 and - (minus sign) and _ (underscore). If id is not specified a new id is generated internally. `required = false`
fileName	string	Filename of the document
format	ERestDocumentFormat	Document format (default PDF). This property can be set to either PDF, MS_WORD, JPEG, PNG, BMP, TIFF or JPG.
signatureFields	Array[RestSignatureField]	A list of all signature fields contained in the DocumentContent. For further details on the signatureField properties refer to the Add Signature Field to Document request.
textFields	Array[RestTextFieldInput]	A list of all text fields contained in the DocumentContent. For further details on the textField properties refer to the Add Text Field to Document request.

 © 2017 Kofax

Key	Value Type	Description
checkboxFields	Array[RestCheckboxFieldInput]	A list of all checkboxes contained in the DocumentContent. For further details on the checkboxField properties refer to the Add Checkbox to Document request.
description	string	Description of the document
name	string	Name of the document
documentMessage	string	Document Message of the document
order	number	Order of the document

Structures RestCheckboxFieldInput, RestTextFieldInput, RestSignatureField are described in chapter Signing Package Properties Overview.

2.3.2 Update a Document

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

NOTE

For this request a valid authentication and User or Signer rights are 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.

NOTE

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:	**/rest/v5/packages/{packageid}/documents/{documentid}**
Producible Formats:	JSON, XML
Return Formats:	JSON, XML
Request Header(s):	Accept: application/json, application/xml
HTTP Method:	PUT

Example Request URL

```
PUT
   https://localhost:8080/cirrus/rest/v5/packages/1001/documents/1002
```

Example Headers

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

Example Request Body (JSON)

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

2.3.2.1 Parameters

Path Parameters

Key	Required	Value Type	Default Value	Description
packageid	yes	string	n/a	ID of the signing package to query the audit trail from
documentid	yes	string	n/a	ID of the document the user wants to update

Body Parameters

Key	Required	Value Type	Default Value	Description
updDocument	yes	RestDocument	null	Represents a RestDocument object either in JSON or XML.

2.3.2.2 Response

The response status is 200 if the document was successfully updated, otherwise a Kofax SignDoc Status Code is returned together with the explaining message(s).

 © 2017 Kofax

2.3.2.3 Fields

Mandatory Fields

Key	Value Type	Description
content	string	Actual document in Base64 format

Available Fields

Key	Value Type	Description
id	string	The id of the document. id must be unique (within one package) and it must consist of characters from A-Z a-z 0-9 and - (minus sign) and _ (underscore). If id is not specified a new id is generated internally. `required = false`
fileName	string	Filename of the document
format	ERestDocumentFormat	Document format (default PDF). This property can be set to either PDF, MS_WORD, JPEG, PNG, BMP, TIFF or JPG.
signatureFields	Array[RestSignatureField]	A list of all signature fields contained in the DocumentContent. For further details on the signatureField properties refer to the Add Signature Field to Document request.
textFields	Array[RestTextFieldInput]	A list of all text fields contained in the DocumentContent. For further details on the textField properties refer to the Add Text Field to Document request.
checkboxFields	Array[RestCheckboxFieldInput]	A list of all checkboxes contained in the DocumentContent. For further details on the checkboxField properties refer to the Add Checkbox to Document request.
description	string	Description of the document
name	string	Name of the document
documentMessage	string	Document Message of the document
order	number	Order of the document

Structures RestCheckboxFieldInput, RestTextFieldInput, RestSignatureField are described in chapter Signing Package Properties Overview.

2.3.3 Delete a Document

Deletes a document and all fields part of the document.

NOTE

For this request a valid authentication and User rights are necessary.

URL:	**/rest/v5/packages/{packageid}/documents/{documentid}**
Producible Formats:	JSON, XML
Return Formats:	JSON, XML
Request Header(s):	Accept: application/json, application/xml
HTTP Method:	DELETE

Example Request URL

```
DELETE
   https://localhost:8080/cirrus/rest/v5/packages/1001/documents/1002
```

Example Headers

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

2.3.3.1 Parameters

Path Parameters

Key	Required	Value Type	Default Value	Description
packageid	yes	string	n/a	ID of the signing package to query the audit trail from
documentid	yes	string	n/a	ID of the document the user wants to delete

 © 2017 Kofax

2.3.3.2 Response

The response status is 200 if the document was successfully deleted, otherwise a Kofax SignDoc Status Code is returned together with the explaining message(s).

2.3.4 Get a Single Document

Returns a single document specified by a given document ID.

NOTE

For this request a valid session is necessary.

For this request a valid authentication and User or Signer rights are 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:	**/rest/v5/packages/{packageid}/documents/{documentid}**
Producible Formats:	JSON, XML
Request Header(s):	Accept: application/json, application/xml
HTTP Method:	GET

Example Request URL

```
GET
   https://localhost:8080/cirrus/rest/v5/packages/1001/documents/1002
```

Example Headers

```
Accept: application/json
```

2.3.4.1 Parameters

Path Parameters

Key	Required	Value Type	Default Value	Description
documentid	yes	string	n/a	ID of the document to query
packageid	yes	string	n/a	ID of the signing package to query the audit trail from

Key	Required	Value Type	Default Value	Description
useIntId	no	boolean	false	The provided ids for package and document are the internal ids which were returned by get audit trail method.

2.3.4.2 Response

The response status is 200 if the document was queried successfully, otherwise a Kofax SignDoc Status Code is returned together with the explaining message(s). In case of a successful query the response body contains the following information regarding the requested document (RestDocumentOutput).

- id
- name
- description
- format (PDF, MS_WORD, JPEG, PNG, BMP, TIFF, JPG)
- content
- order
- documentMessage
- thumbnail
- fileName
- signatureFields (a list of RestSignatureFieldOutput)
  - id
  - name
  - readOnly
  - required
  - label
  - signerId
  - page
  - posx
  - posy
  - width
  - height
  - signingModeOptions
  - tabIndex
  - state (PENDING, SIGNED)
- textFields (a list of RestTextFieldOutput)
  - id

 © 2017 Kofax

    - name
    - required
    - label
    - signerId
    - readOnly
    - page
    - posx
    - posy
    - width
    - height
    - data
    - multipleLine
    - tabIndex
    - maxLength
- checkboxFields (a list of RestCheckboxFieldOutput)
    - id
    - name
    - readOnly
    - required
    - label
    - signerId
    - page
    - posx
    - posy
    - width
    - height
    - tabIndex
    - state (UNCHECKED, CHECKED)

2.4 Event Requests

Trigger an Event

2.4.1 Trigger an Event

Trigger an event to the system.

NOTE

For this request a valid authentication and User or Signer rights are necessary.

URL:	**/rest/v5/event**
Producible Formats:	JSON, XML
Request Header(s):	Accept: application/json, application/xml
HTTP Method:	POST

Example Request URL

```
POST https://localhost:8080/cirrus/rest/v5/event
```

Example Headers

```
Accept: application/json
```

Example Request Body (JSON)

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

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

Key	Required	Type	Description
action	yes	string	Action name which should be triggered. It can be applied to specified subject.
subject	no	string	Subject for specified action.
product	no	string	Possible value: CIRRUS. If not specified, the event should be handled by Kofax SignDoc.

 © 2017 Kofax

Structure of the event input object:

Structure	Element	Element Type	Description
RestEvent	list	array[RestEntry]	List of properties (as key-value pairs) of the event
RestEntry	k	string	The key of the pair
	v	string	The value of the pair

There is the following list of events, which can be processed by Kofax SignDoc:

Action	Subject	Product	Parameters	Description
END	SIGNING_SESSION	CIRRUS	n/a	End the existing Common Signing Session. Signing host can finish the signing session any time triggering the event. The signing session will be reset, authentication token will be invalidated. Note: can be triggered by a signing host only.
	SIGNER	CIRRUS	n/a	End a package signing by a signer. The signer has to complete signing/reviewing of all required documents before sending the event. The signing package owner will be notified, the proper message will be added to the audit trail. Note: can be triggered by a signer only.
AGREE_ESIGN_CONSENT	SIGNER	CIRRUS	n/a	Accept the e-sign consent by an authenticated signer. The consent must be accepted once by each signer. The proper message will be added to the audit trail. Note: can be triggered by a signer only.
DECLINE	SIGNER	CIRRUS	DECLINE_REASON (mandatory) DECLINE_COMMENT (optional)	Decline a signing package by a signer. The proper message will be added to the audit trail. Note: can be triggered by a signer only.
DELIVERED	SIGNER	CIRRUS	DOCUMENT_ID (mandatory)	A document page was delivered to the signer. The proper message will be added to the audit trail.

Action	Subject	Product	Parameters	Description
			PAGE_NUMBER (mandatory)	Note: can be triggered by a signer only.
DONE	SIGNER_DOCUMENT	CIRRUS	n/a	A signer has completed a document. The proper message will be added to the audit trail. Note: can be triggered by a signer only.

2.4.1.1 Parameters

Body Parameters

Key	Required	Value Type	Default Value	Description
restEvent	yes	RestEvent	n/a	Represents a RestEvent object either in JSON or XML.

2.4.1.2 Response

It will return OK (200) after the event has been triggered to the system successfully. If the specified event or its parameters are not valid, proper error message will be returned.

2.5 Field Requests

Add Checkbox to Document
Get a Single Checkbox
Update a Checkbox with Form Data
Get All Fields
Delete a Field
Add Signature Field to Document
Get a Single Signature Field
Update a Signature Field with Form Data
Add Text Field to Document
Update a Signature Field with Form Data
Get a Single Text Field

 © 2017 Kofax

2.5.1 Add Checkbox to Document

Adds a new checkbox to an existing document. An input JSON or XML string specifying the details of the signature field has to be provided as a request parameter. The specification has to provide at least xpos, ypos, width, height, label, and fieldname. 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 checkbox have to be in the bounds of the document.

NOTE

For this request a valid session and User rights are necessary.

URL:	**/rest/v5/packages/{packageid}/documents/ {documentid}/checkbox**
Producible Formats:	JSON, XML
Return Formats:	JSON, XML
Request Header(s):	Accept: application/json, application/xml
HTTP Method:	POST

Example Request URL

```
POST
   https://localhost:8080/cirrus/rest/v5/packages/1001/documents/1002
   /checkbox
```

Example Headers

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

Example Request Body (JSON)

```
{
  "state": "CHECKED",
  "readOnly": false,
  "required": false,
  "id": "checkbox1",
  "posy": 10,
  "posx": 10,
  "width": 200,
  "height": 100,
  "fieldName": "field1",
  "label": "field1",
  "page": 1
}
```

2.5.1.1 Parameters

Path Parameters

Key	Required	Value Type	Default Value	Description
packageid	yes	string	n/a	ID of the signing package to query the audit trail from
documentid	yes	string	n/a	ID of the document the user wants to add a checkbox

Body Parameters

Key	Required	Value Type	Default Value	Description
checkBox	yes	RestCheckBoxInput	null	Represents a RestCheckBoxInput object either in JSON or XML.

2.5.1.2 Fields

Mandatory Fields

Key	Value Type	Description
id	string	ID of the field
page	integer	The page of the field
posx	integer	The posx of the field
posy	integer	The posy of the field
width	integer	The width of the field
height	integer	The height of the field

Available Fields

Key	Value Type	Description
id	string	The id of the field
signerId	string	The signer (identifier) assigned to the field
label	string	The label of the field

 © 2017 Kofax

Key	Value Type	Description
required	boolean	Whether the field is required or not
page	integer	The page of the field
posx	integer	The posx of the field
posy	integer	The posy of the field
width	integer	The width of the field
height	integer	The height of the field
readOnly	boolean	Whether the field is readonly or not
state	CheckboxFieldState	The state of the checkbox. This property can be set to either UNCHECKED or CHECKED.

2.5.2 Get a Single Checkbox

Returns a single checkbox specified by a given checkbox ID.

NOTE

For this request a valid authentication and User rights are necessary.

URL:	**/rest/v5/packages/{packageid}/documents/{documentid}/checkboxes/{checkboxid}**
Producible Formats:	JSON, XML
Request Header(s):	Accept: application/json, application/xml
HTTP Method:	GET

Example Request URL

```
GET
   https://localhost:8080/cirrus/rest/v5/packages/1001/documents/1002
   /checkboxes/1003
```

Example Headers

```
Accept: application/json
```

Example Response Body (JSON)

```
{
  "id": 403,
  "fieldName": "cbox-1",
  "label": "checkbox 1",
  "required": false,
  "readOnly": false,
  "page": 1,
  "posx": 108,
  "posy": 345,
  "width": 20,
  "height": 20,
  "creationTime": 1418821332015,
  "state": "UNCHECKED"
}
```

2.5.2.1 Parameters

Path Parameters

Key	Required	Value Type	Default Value	Description
packageid	yes	string	n/a	ID of the signing package to query the audit trail from
documentid	yes	string	n/a	ID of the document the user wants to add a check box
checkboxid	yes	string	n/a	ID of the checkbox to query

Query Parameters

Key	Required	Value Type	Default Value	Description
useIntId	no	boolean	false	The provided ids for package, document and field are the internal ids which were returned by get audit trail method.

2.5.2.2 Response

The response status is 200 if the checkbox was queried successfully, otherwise a Kofax SignDoc Status Code is returned together with the explaining message(s).

2.5.3 Update a Checkbox with Form Data

This request updates an existing checkbox identified by ID with form data. Please note that the page, coordinates and dimensions for the update have to be in bounds of the document.

 © 2017 Kofax

NOTE

For this request a valid authentication and User rights are necessary.

URL:	**/rest/v5/packages/{packageid}/documents/{documentid}/checkboxes/{checkboxid}**
Producible Formats:	JSON, XML
Request Header(s):	Accept: application/json, application/xml
HTTP Method:	POST

Example Request URL

```
POST
   https://localhost:8080/cirrus/rest/v5/packages/1001/documents/1002
   /checkboxes/1003
```

Example Headers

```
Accept: application/json
Content-Type: multipart/form-data
```

2.5.3.1 Parameters

Path Parameters

Key	Required	Value Type	Default Value	Description
packageid	yes	string	n/a	ID of the signing package to query the audit trail from
documentid	yes	string	n/a	ID of the document the user wants to add a check box
checkboxid	yes	string	n/a	ID of the checkbox to update

Query Parameters

Key	Required	Value Type	Default Value	Description
fieldName	no	string	n/a	New fieldName of checkbox
label	no	string	n/a	New label of checkbox
required	no	boolean	n/a	New required of checkbox

Key	Required	Value Type	Default Value	Description
page	no	integer	n/a	New page of checkbox
posx	no	integer	n/a	New posx of checkbox
posy	no	integer	n/a	New posy of checkbox
width	no	integer	n/a	New width of checkbox
height	no	integer	n/a	New height of checkbox
state	no	string	n/a	New state of checkbox
readOnly	no	boolean	n/a	New readOnly of checkbox
signerid	no	integer	n/a	ID of a signer. The specified signer will be assigned as the new signer of the checkbox.

2.5.3.2 Fields

Available Fields

Key	Value Type	Description
id	string	The id of the field
signerId	string	The signer (identifier) assigned to the field
label	string	The label of the field
required	boolean	Whether the field is required or not
page	integer	The page of the field
posx	integer	The posx of the field
posy	integer	The posy of the field
width	integer	The width of the field
height	integer	The height of the field
readOnly	boolean	Whether the field is readonly or not
state	CheckboxFieldState	The state of the checkbox. This property can be set to either UNCHECKED or CHECKED.

 © 2017 Kofax

2.5.3.3 Response

The response status is 200 if the checkbox was successfully updated, otherwise a Kofax SignDoc Status Code is returned together with the explaining message(s).

2.5.4 Get All Fields

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

NOTE

For this request a valid authentication and User rights are necessary.

URL:	**/rest/v5/packages/{packageid}/documents/{documentid}/fields**
Producible Formats:	JSON, XML
Request Header(s):	Accept: application/json, application/xml
HTTP Method:	GET

Example Request URL

```
GET
   https://localhost:8080/cirrus/rest/v5/packages/1001/documents/1002
   /fields
```

Example Headers

```
Accept: application/json
```

Example Response Body (JSON)

```
[
  {
    "id": 403,
    "name": "cbox-1",
    "type": "CheckBox",
    "signerID": 251,
    "url": "/rest/v5/packages/documents/checkboxes/403"
  },
  {
    "id": 401,
    "name": "signature-1",
    "type": "SignatureField",
    "signerID": 251,
    "url": "/rest/v5/packages/documents/signaturefields/401"
  },
  {
    "id": 402,
    "name": "text-1",
    "type": "TextField",
    "signerID": 251,
    "url": "/rest/v5/packages/documents/textfields/402"
  }
]
```

2.5.4.1 Parameters

Path Parameters

Key	Required	Value Type	Default Value	Description
packageid	yes	string	n/a	ID of the signing package to query the audit trail from
documentid	yes	string	n/a	ID of the document the user wants to get the fields from.

Query Parameters

Key	Required	Value Type	Default Value	Description
fieldFilter	no	ERestDocument FieldType	n/a	Whether the returned list should be filtered to a specific field type or not.

2.5.4.2 Response

The response status is 200 if list was queried successfully, otherwise a Kofax SignDoc Status Code is returned together with the explaining message(s).

2.5.5 Delete a Field

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

NOTE

For this request a valid authentication and User rights are necessary.

URL:	**/rest/v5/packages/{packageid}/documents/ {docuemntid}/fields/{fieldid}**
Producible Formats:	JSON, XML
Return Formats:	JSON, XML
Request Header(s):	Accept: application/json, application/xml

 © 2017 Kofax

HTTP Method:	DELETE

Example Request URL

```
DELETE
   https://localhost:8080/cirrus/rest/v5/packages/1001/documents/1002
   /fields/1003
```

Example Headers

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

2.5.5.1 Parameters

Path Parameters

Key	Required	Value Type	Default Value	Description
fieldid	yes	string	n/a	ID of the field the user wants to delete
documentid	yes	string	n/a	ID of the document to query
packageid	yes	string	n/a	ID of the signing package to query the audit trail from

2.5.5.2 Response

The response status is 200 if the field was successfully deleted, otherwise a Kofax SignDoc Status Code is returned together with the explaining message(s).

2.5.6 Add Signature Field to Document

Adds a new signature field to an existing document. An input JSON or XML string specifying the details of the signature field has to be provided as a request parameter. The specification has to provide at least xpos, ypos, width, height, label, and fieldname.

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 PH, HW and C2S.

Please note that the page, coordinates and dimensions of the new signature field have to be in bounds of the document.

NOTE

For this request a valid session and User rights are necessary.

URL:	**/rest/v5/packages/{packageid}/documents/{documentid}/signaturefield**
Producible Formats:	JSON, XML
Return Formats:	JSON, XML
Request Header(s):	Accept: application/json, application/xml
HTTP Method:	POST

Example Request URL

```
POST
   https://localhost:8080/cirrus/rest/v5/packages/1001/documents/1002
   /signaturefield
```

Example Headers

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

Example Request Body (JSON)

```
{
  "readOnly": false,
  "required": false,
  "posy": 10,
  "posx": 10,
  "width": 200,
  "height": 100,
  "label": "field1",
  "page": 1
}
```

2.5.6.1 Parameters

Path Parameters

Key	Required	Value Type	Default Value	Description
packageid	yes	string	n/a	ID of the signing package to query the audit trail from
documentid	yes	string	n/a	ID of the document the user wants to add a signature field

 © 2017 Kofax

Body Parameters

Key	Required	Value Type	Default Value	Description
signaturefield	yes	RestSignatureFieldInput	null	Represents a RestSignatureFieldInput object either in JSON or XML.

2.5.6.2 Response

The response status is 201 (CREATED) if the signature field was created, otherwise a Kofax SignDoc Status Code is returned together with the explaining message(s).

2.5.6.3 Fields

Mandatory Fields

Key	Value Type	Description
id	string	The id of the field
page	integer	The page of the field
posx	integer	The posx of the field
posy	integer	The posy of the field
width	integer	The width of the field
height	integer	The height of the field

Available Fields

Key	Value Type	Description
id	string	The id of the field
signerId	string	The signer (identifier) assigned to the field
label	string	The label of the field
required	boolean	Whether the field is required or not
page	integer	The page of the field
posx	integer	The posx of the field
posy	integer	The posy of the field

Key	Value Type	Description
width	integer	The width of the field
height	integer	The height of the field
readOnly	boolean	Whether the field is readonly or not
signingModeOptions		A list of the signing modes which should be allowed for the signer, can be: HW - for handwritten (Tablet or HTML5) PH - photo, captured by camera C2S - Click2Sign Default: HW, PH and C2S

2.5.7 Get a Single Signature Field

Returns a single signature field specified by a given signature field ID.

NOTE

For this request a valid authentication and User rights are necessary.

URL:	**/rest/v5/packages{packageid}/documents/{documentid}/signaturefields/{signaturefieldid}**
Producible Formats:	JSON, XML
Request Header(s):	Accept: application/json, application/xml
HTTP Method:	GET

Example Request URL

```
GET
   https://localhost:8080/cirrus/rest/v5/packages/1001/documents/1002
   /signaturefields/1003
```

Example Headers

```
Accept: application/json
```

Example Response Body (JSON)

 © 2017 Kofax

```
{
  "id": 401,
  "fieldName": "signature-1",
  "label": "signature field 1",
  "required": true,
  "readOnly": false,
  "page": 1,
  "posx": 151,
  "posy": 108,
  "width": 200,
  "height": 64,
  "creationTime": 1418821325736,
  "state": "PENDING",
  "signingModeOptions": [
    "HW",
    "PH",
    "C2S"
  ]
}
```


2.5.7.1 Parameters

Path Parameters

Key	Required	Value Type	Default Value	Description
packageid	yes	string	n/a	ID of the signing package to query the audit trail from
documentid	yes	string	n/a	ID of the document the user wants to add a signature field
signaturefieldid	yes	string	n/a	ID of the signature field to query

Query Parameters

Key	Required	Value Type	Default Value	Description
useIntId	no	boolean	false	The provided ids for package, document and field are the internal ids which were returned by get audit trail method.

2.5.7.2 Response

The response status is 200 if the signature field was queried successfully, otherwise a Kofax SignDoc Status Code is returned together with the explaining message(s).

2.5.8 Update a Signature Field with Form Data

This request updates an existing signature field identified by ID with form data. Please note that the page, coordinates and dimensions for the update have to be in bounds of the document.

NOTE

For this request a valid authentication and User rights are necessary.

URL:	**/rest/v5/packages/{packageid}/documents/{documentid}/signaturefields/{signaturefieldid}**
Producible Formats:	JSON, XML
Request Header(s):	Accept: application/json, application/xml
HTTP Method:	POST

Example Request URL

```
POST
   https://localhost:8080/cirrus/rest/v5/packages/1001/documents/1002
   /signaturefields/1003
```

Example Headers

```
Accept: application/json
Content-Type: multipart/form-data
```

2.5.8.1 Parameters

Path Parameters

Key	Required	Value Type	Default Value	Description
packageid	yes	string	n/a	ID of the signing package to query the audit trail from
documentid	yes	string	n/a	ID of the document the user wants to add a signature field
signaturefieldid	yes	string	n/a	ID of the signature field to update

 © 2017 Kofax

Query Parameters

Key	Required	Value Type	Default Value	Description
fieldName	no	string	n/a	New fieldid of signature field
label	no	string	n/a	New label of signature field
required	no	boolean	n/a	New required status of signature field
page	no	integer	n/a	New page of signature field
posx	no	integer	n/a	New posx of signature field
posy	no	integer	n/a	New posy of signature field
width	no	integer	n/a	New width of signature field
height	no	integer	n/a	New height of signature field
signerid	no	integer	n/a	ID of a signer. The specified signer will be assigned as the new signer of the signature field.

2.5.8.2 Fields

Available Fields

Key	Value Type	Description
id	string	The id of the field
signerId	string	The signer (identifier) assigned to the field
label	string	The label of the field
required	boolean	Whether the field is required or not
page	integer	The page of the field
posx	integer	The posx of the field
posy	integer	The posy of the field
width	integer	The width of the field
height	integer	The height of the field
readOnly	boolean	Whether the field is readonly or not

Key	Value Type	Description
signingModeOptions		A list of the signing modes which should be allowed for the signer, can be: HW - for handwritten (Tablet or HTML5) PH - photo, captured by camera C2S - Click2Sign Default: HW, PH and C2S

2.5.8.3 Response

The response status is 200 if the signature field was successfully updated, otherwise a Kofax SignDoc Status Code is returned together with the explaining message(s).

2.5.9 Add Text Field to Document

Adds a new text field to an existing document. An input JSON or XML string specifying the details of the signature field, has to be provided as a request parameter. The specification has to provide at least xpos, ypos, width, height, label, and fieldname.

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 text field have to be in bounds of the document.

NOTE

For this request a valid session and User rights are necessary.

URL:	**/rest/v5/packages/{packageid}/documents/{documentid}/textfield**
Producible Formats:	JSON, XML
Return Formats:	JSON, XML
Request Header(s):	Accept: application/json, application/xml
HTTP Method:	POST

Example Request URL

```
POST
   https://localhost:8080/cirrus/rest/v5/packages/1001/documents/1002
   /textfield
```

 © 2017 Kofax

Example Headers

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

Example Request Body (JSON)

```
{
  "data": "data",
  "maxLength": 100,
  "multipleLine": false,
  "readOnly": false,
  "required": false,
  "posy": 10,
  "posx": 10,
  "width": 200,
  "height": 100,
  "fieldName": "field1",
  "label": "field1",
  "page": 1
}
```

2.5.9.1 Parameters

Path Parameters

Key	Required	Value Type	Default Value	Description
packageid	yes	string	n/a	ID of the signing package to query the audit trail from
documentid	yes	string	n/a	ID of the document the user wants to add a text field

Body Parameters

Key	Required	Value Type	Default Value	Description
textfield	yes	RestTextFieldInput	null	Represents a RestTextFieldInput object either in JSON or XML.

2.5.9.2 Response

The response status is 201 (CREATED) if the text field was created, otherwise a Kofax SignDoc Status Code is returned together with the explaining message(s).

2.5.9.3 Fields

Mandatory Fields

Key	Value Type	Description
id	string	The id of the field
page	integer	The page of the field
posx	integer	The posx of the field
posy	integer	The posy of the field
width	integer	The width of the field
height	integer	The height of the field

Available Fields

Key	Value Type	Description
id	string	The id of the field
signerId	string	The signer (identifier) assigned to the field
label	string	The label of the field
required	boolean	Whether the field is required or not
page	integer	The page of the field
posx	integer	The posx of the field
posy	integer	The posy of the field
width	integer	The width of the field
height	integer	The height of the field
readOnly	boolean	Whether the field is readonly or not
data	string	The data of the field
multipleLine	boolean	Whether the field allows multiple lines or not
maxLength	Integer	The max lenght of the field

 © 2017 Kofax

2.5.10 Update a Text Field with Form Data

This request updates an existing text field identified by ID with form data. Please note that the page, coordinates and dimensions for the update have to be in bounds of the document.

NOTE

For this request a valid API key and User rights are necessary.

URL:	**/rest/v5/packages/{packageid}/documents/{documentid}/textfields/{textfieldid}**
Producible Formats:	JSON, XML
Request Header(s):	Accept: application/json, application/xml
HTTP Method:	POST

Example Request URL

```
POST
   https://localhost:8080/cirrus/rest/v5/packages/1001/documents/1002
   /textfields/1003
```

Example Headers

```
Accept: application/json
Content-Type: multipart/form-data
```

2.5.10.1 Parameters

Path Parameters

Key	Required	Value Type	Default Value	Description
packageid	yes	string	n/a	ID of the signing package to query the audit trail from
documentid	yes	string	n/a	ID of the document the user wants to add a text field
textfieldid	yes	string	n/a	ID of the text field to update

Query Parameters

Key	Required	Value Type	Default Value	Description
fieldName	no	string	n/a	New fieldName of text field
label	no	string	n/a	New label of text field
required	no	boolean	n/a	New required of text field
page	no	integer	n/a	New page of text field
posx	no	integer	n/a	New posx of text field
posy	no	integer	n/a	New posy of text field
width	no	integer	n/a	New width of text field
height	no	integer	n/a	New height of text field
data	no	string	n/a	New data of text field
multipleLine	no	boolean	n/a	New multipleLine of text field
readOnly	no	boolean	n/a	New readOnly of text field
maxLength	no	integer	n/a	New maxLength of text field
signerid	no	integer	n/a	ID of a signer. The specified signer will be assigned as the new signer of the signature field.

2.5.10.2 Fields

Available Fields

Key	Value Type	Description
id	string	The id of the field
signerId	string	The signer (identifier) assigned to the field
label	string	The label of the field
required	boolean	Whether the field is required or not
page	integer	The page of the field
posx	integer	The posx of the field

 © 2017 Kofax

Key	Value Type	Description
posy	integer	The posy of the field
width	integer	The width of the field
height	integer	The height of the field
readOnly	boolean	Whether the field is readonly or not
data	string	The data of the field
multipleLine	boolean	Whether the field allows multiple lines or not
maxLength	Integer	The max lenght of the field

2.5.11 Get a Single Text Field

Returns a single text field specified by a given text field ID.

NOTE

For this request a valid authentication and User rights are necessary.

URL:	**/rest/v5/packages/{packageid}/documents/{documentid}/textfields/{textfieldid}**
Producible Formats:	JSON, XML
Request Header(s):	Accept: application/json, application/xml
HTTP Method:	GET

Example Request URL

```
GET
   https://localhost:8080/cirrus/rest/v5/packages/1001/documents/1002
   /textfields/1003
```

Example Headers

```
Accept: application/json
```

Example Response Body (JSON)


```
{
  "id": 402,
  "fieldName": "text-1",
  "label": "textfield field 1",
  "required": true,
  "readOnly": false,
  "page": 1,
  "posx": 81,
  "posy": 241,
  "width": 300,
  "height": 30,
  "lastUpdateTime": 1418821330411,
  "creationTime": 1418821329503,
  "data": "",
  "multipleLine": false
}
```


2.5.11.1 Parameters

Path Parameters

Key	Required	Value Type	Default Value	Description
packageid	yes	string	n/a	ID of the signing package to query the audit trail from
documentid	yes	string	n/a	ID of the document the user wants to add a text field
textfieldid	yes	string	n/a	ID of the text field to query

Query Parameters

Key	Required	Value Type	Default Value	Description
useIntId	no	boolean	false	The provided ids for package, document and field are the internal ids which were returned by get audit trail method.

2.5.11.2 Response

The response status is 200 if the text field was successfully updated, otherwise a Kofax SignDoc Status Code is returned together with the explaining message(s).

 © 2017 Kofax

2.6 Signing Package Requests

Create a New Signing Package
Complete Signing Package Structure
Delete a Signing Package
Update a Signing Package
Update an Existing Signature Field with Form Data
Get a Single Signing Package
Get Audit Trail
Get a List of Packages
Schedule a Signing Package
Prepare Signing Session

2.6.1 Create a New Signing Package

It is possible to create a signing package from scratch or as copy of another already existing package.

If you want to create a signing package on base of another package you have to define either the source package identifier (query parameter) src_id, which is returned as id in a RestPackageListEntry object returned by a Get a List of Packages request.

In order to create a signing package from scratch, an input JSON or XML string specifying the details of the signing package, has to be provided in a request body. The specification has to provide at least a package name and a signing package type (PACKAGE, TEMPLATE). This request prepares a default signing package if nothing further is specified.

To immediately add documents to the signing package a RestDocument object has to be added (see Example Body of the documentation).

To immediately assign a signer to a field use the id attribute of the signer to make a cross reference (see Example Body of the documentation).

In Example Body a signer with id "signer1" is created and later cross-referenced in the signer attribute of the signature field. The signing package is still created without assigning a signer to the signature field, if the cross reference is incorrect.

NOTE

For this request a valid authentication and User rights are necessary.

If schedule=true is set as query parameter then the signing package is scheduled directly after successful creation if all criteria for scheduling are satisfied.

URL:	**/rest/v5/package**
Producible Formats:	JSON, XML
Input Formats:	JSON, XML

Request Header(s):	Accept: application/json, application/xml
HTTP Method:	POST

Example Request URL

```
POST https://localhost:8080/cirrus/rest/v5/package
```

Example Headers

```
Accept: application/json
```

2.6.1.1 Parameters

Query Parameters

Key	Required	Value Type	Default Value	Description
src_id	no	string	n/a	ID of the source package which is used as base for the new package
schedule	no	bool	n/a	Schedules the Signing Package immediately after creation if all start criteria are fulfilled
delete_existing	no	bool	n/a	Deletes existing package with the same id before creating a new package
autoprepare	no	bool	n/a	Whether the Signing Package should be auto prepared or not

Body Parameters

Key	Required	Value Type	Default Value	Description
restSigningPackageInput	yes	RestSigningPackageInput	n/a	Represents a RestSigningPackageInput object either in JSON or XML.

2.6.1.2 Response

The response status is 201 if the signing package was successfully added, otherwise a Kofax SignDoc Status Code is returned together with the explaining message(s).

 © 2017 Kofax

2.6.1.3 Fields

Mandatory Fields

Key	Value Type	Description
n/a	no	n/a

Available Fields

Key	Value Type	Description
id	string	The id of the package
name	string	The name of the package
type	string	The type of the package. This property can be set to either PACKAGE or TEMPLATE
description	string	The description of the package
processingType	string	The processing type of the package. This property can be set to either SEQ or PAR.
expirationDate	string	The expiration date of the package. Format: date-time with a time-zone in UTC, such as '2007-12-03T10:15:30Z' (ISO-8601).
startDate	string	The start date of the package. Format: date-time with a time-zone in UTC, such as '2007-12-03T10:15:30Z' (ISO-8601).
mailSubject	string	The mail subject of the package
mailMessage	string	The mail message of the package
auditTrailOptions	number	The audit trail options of the package
signers	Array[RestSignerInput]	A list of all signers contained in the package
documents	Array[RestDocumentInput]	A list of all documents contained in the package
reminders	Array[RestReminderInput]	A list of all reminders contained in the package
signingModeOption	Array[String]	A list of the default signing modes. This property can be set to either PH, HW or C2S.

2.6.1.4 Example Response Body

Example Response Body (JSON)

```
{
  "id": "66d3b61b-7125-40cb-ac00-fd186a9c6243"
}
```

Example Response Body (XML)

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<RestID>
  <ID>1a855f5d-abe1-43bd-89f4-7238a8f00316</ID>
</RestID>
```

2.6.1.5 Example Body

Example Body (JSON)

```
{
  "name": "package1",
  "type": "PACKAGE",
  "processingType": "SEQ",
  "signers": [
     {
     "order": 0,
     "id": "signer1",
     "email": "signer1@kofax.com",
     "role": "SIGNER",
     "name": "Signer1"
     }
  ],
  "documents": [
     {
     "name":"Rest Test",
     "order":0,
     "description":"Rest Test",
     "documentMessage":"Please",
     "fileName":"RestTest1.pdf",
     "signatureFields": [
       {
       "readOnly":"false",
       "required":"false",
       "posy":35,
       "posx":227,
       "width":281,
       "height":92,
       "id":"Applicant",
       "page":1,
       "signerId": "signer1",
       "label":"Applicant"
       }
   ],
       "content":  "base64 encoded PDF"
    }
    "esignConsent": "Text of e-sign consent"
  ]
}
```

2.6.1.6 Create Signing Package Detailed Scenarios

To create a signing package via the Kofax SignDoc REST API a user has several different options.

 © 2017 Kofax

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 or by creating a signing package from an already existing template.

It additionally is possible to create a signing package by using a Word document with already aligned signature lines.

The following chapter will cover the most important approaches on how to create a signing package via the Create Signing Package request.

2.6.1.6.1 Parameters Overview

The following parameters can be set for the Create Signing Package request.

2.6.1.6.1.1 Query Parameters

Key	Required	Value Type	Default Value	Description
src_id	no	string	n/a	ID of the source package which is used as base for the new package.
schedule	no	bool	false	Schedules the Signing Package immediately after creation if all start criteria are fulfilled.
delete_existing	no	bool	false	Deletes existing package with the same id before creating a new package.
autoprepare	no	bool	false	Whether the Signing Package should be auto prepared or not.

2.6.1.6.1.2 Body Parameters

Key	Required	Value Type	Default Value	Description
restSigningPackageInput	yes	RestSigningPackageInput	n/a	Represents a RestSigningPackageInput object either in JSON or XML.

2.6.1.6.1.3 Parameters in More Detail

The **src_id** query parameter specifies the ID of a template, a new Signing Package is supposed to be based on. This parameter enables a complete different approach to create a Signing Package and uses a Signing Package specified in the body parameter purely as metadata. A Template usually has all Documents Signers and fields predefined. And a new Signing Package created with this option is an exact copy of this Template. 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 please consult Scenario 3 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 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 Singing Packages with the same ID are not allowed within Kofax SignDoc. 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. Kofax SignDoc does delete any Signing Package with the same ID before creating the new Singing Package in one request. If no ID is specified in the Signing Package data structure the query parameter delete_existing is ignored.

The **autoprepare** parameter is used to automatically prepare a specified document, by adding Signer and Fields to the Document.

The **body** parameter is used to specify a complete Signing Package as either a JSON or XML structure. A new Singing 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 please consult Scenario 3 of this chapter.

2.6.1.6.1.4 Parameter Use Cases

Parameter	Create Signing Package from scratch	Create Signing Package from template	Create Signing Package with Word Document
src_id	no	yes	no
schedule	yes	yes	yes
delete_existing	yes	yes	yes
autoprepare	yes	yes	yes

2.6.1.6.2 Signing Package Properties Overview

The following table specifies the possible properties a user can set in the body parameter.

Every property required to create a signing package is declared as:

```
required = true
```

Class	Element	Element Type	Description
RestSigningPack	id	string	The id of the package.

 © 2017 Kofax

Class	Element	Element Type	Description
ageInput			The id of the package is required for identification if query parameter delete_existing is set to true otherwise id is optional. The id is also stored in the DB and can be used for later identification. The package id must be unique (within one account) and it must consist of characters from A-Z a-z 0-9 and - (minus sign) and _ (underscore). If id is not specified a new id is generated internally.
	name	string	The name of the package. `required = false`
	type	SigningPackageType	The type of the package. Possible values: PACKAGE, TEMPLATE `required = false`
	description	string	The description of the package. `required = false`
	processingType	ProcessingType	The processing type of the package. Possible values: SEQ, PAR `required = false`
	startDate	LocalDateTime	The expiration date of the package. Format: date-time with a time-zone in UTC, such as '2007-12-03T10:15:30Z' (ISO-8601).. `required = false`
	expirationDate	LocalDateTime	The start date of the package. Format: date-time with a time-zone in UTC, such as '2007-12-03T10:15:30Z' (ISO-8601).. `required = false`
	mailSubject	string	The mail subject of the package `required = false`
	mailMessage	string	The mail message of the package `required = false`
	auditTrailOptions	int	The audit trail options of the package. `required = false`

Class	Element	Element Type	Description
	signers	List<RestSignerInput>	The list of all signers contained in the package `required = false`
	documents	List<RestDocumentInput>	The list of all documents contained in the package. `required = false`
	reminders	List<RestReminderInput>	The list of reminders contained in the package `required = false`
	signingModeOptions	List<SigningMode>	A list of the signing modes which are used as default Signing Modes for each Signer that is created within the actual "Create Signing Package" call. Specific signingModeOptions within RestSignatureFieldInput overwrites these default settings for the field. See description of signingModeOptions in RestSignatureFieldInput for more information about the values. `required = false`
RestDocumentInput	id	string	The id of the document. id must be unique (within one package) and it must consist of characters from A-Z a-z 0-9 and - (minus sign) and _ (underscore). If id is not specified a new id is generated internally. `required = false`
	filename	string	The filename of the document. `required = false`
	content	string	The content of the document (Base64 encoded), can be PDF (default), an image (currently JPEG, TIFF, PNG and BMP) or a MS Word document. Default: PDF `required = true`
	order	short	The order of the document in package's sequence of documents. `required = false`

 © 2017 Kofax

Class	Element	Element Type	Description
	signatureFields	List<RestSignatureFieldInput>	A list of all signature fields contained in the document `required = false`
	textFields	List<RestTextFieldInput>	A list of all text fields contained in the document `required = false`
	checkboxFields	List<RestCheckboxFieldInput>	A list of all checkbox fields contained in the document `required = false`
	format	ERestDocumentFormat	The format of the document content, can be PDF, JPEG, TIFF, PNG, BMP or MS_WORD Default: PDF `required = false`
	name	string	The name of the document `required = true`
	description	string	The description of the document `required = false`
	documentMessage	string	Optional message from the sender that is displayed when the recipient opens the document `required = false`
RestSignerInput	id	string	The id of the signer which must be unique within the package. It is necessary to provide an id for referencing a signer in a document field as signerId `required = false`
	name	string	The name of the signer (will be used as form of address in the email, example: Dear Jon Doe, ...) `required = false`
	email	string	The email address of the signer `required = false`
	role	SignerRole	Signer role, can be SIGNER or REVIEWER. Default: SIGNER `required = false`

Class	Element	Element Type	Description
	order	short	The order of the signer. Used when package's processingType is SEQ. `required = false`
	reasonForDecline	SignerReasonForDecline	The reason for decline of the signer. Possible values: • 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 e-sign consent) `required = false`
	commentForDecline	string	The comment for decline of the signer. `required = false`
	authenticationMode	ERestAuthenticationMode	The authentication mode for the signer. Possible values: • NONE (no additional authentication required) • CODE (access code must be provided by the sender) `required = false`
	authenticationParam	string	The authentication code for the signer. `required = false`
	accessCodeDeliveryPluginId	string	The id of the plugin for the access code delivery
	accessCodeDeliveryParameters	List<RestAcessCodeDeliveryChannelParameter>	The list of parameters of the plugin for the access code delivery
	clientCertificateRequest	ERestClientCertificateRequest	The setting specifies whether to use client certificate for signing. It is relevant only when account-specific configuration setting 'client.signing.certificate.request' is set to OPTIONAL. Possible values: • DISABLED • REQUIRED

 © 2017 Kofax

Class	Element	Element Type	Description
	tspSignerInfo	RestTSPSignerInfoInput	TSP-related info for the signer to sign documents using a Trusted Service Provider (TSP), including user's credentials to the TSP and list of documents required signing by TSP. `required = false`
RestTSPSignerInfoInput	tspPluginId	string	The Id of the plugin needed for an interaction with the required TSP. The plugin must be available for the account. `required = true`
	tspSignatureType	ERestTSPSignatureType	The type of signature which will be applied by the required TSP. Possible values: • BASIC • ADVANCED • QUALIFIED If it's not specified, the TSP signature type will be retrieved from account-specific configuration setting 'client.signing.tsp.signature.type' `required = false`
	tspCredentials	List<RestEntry>	The list of credentials needed for an authentication in the required TSP. `required = true`
	tspSignatureDocuments	List<String>	List of IDs of documents needed to be signed by the signer using the required TSP. At least one document must be specified to start the signing of the package. `required = true`
RestEntry	k	string	The key of the entry `required = true`
	v	string	The value of the entry `required = false`
RestTSPSignatureDocument	documentId	string	Id of the document
	state	ERestTSPRequest	Current state of the document in TSP signing process. Possible values:

Class	Element	Element Type	Description
			• REQUIRED • DONE
RestAcessCode DeliveryChannel Parameter	key	string	The key of the parameter
	value	string	The value of the parameter
RestSignatureFieldInput	id	string	Id of the field (used as fieldname in PDF document) which must be unique within the document `required = false`
	signerId	string	The id of the signer (id from RestSignerInput) which is assigned to the field `required = false`
	name	string	The name of the document's field. `required = false`
	label	string	The label of the field `required = true`
	required	boolean	Specifies whether the field is mandatory for the signer or not Default: false `required = false`
	signed	boolean	Specifies whether the field is signed or not Default: false `signed = false`
	readOnly	boolean	Specifies whether the field is read only or not Default: false `required = false`
	height	integer	The field height `required = true`
	width	integer	The field width `required = true`
	page	integer	The page number (starting with 1) `required = true`

 © 2017 Kofax

Class	Element	Element Type	Description
	posx	integer	x-coordinate of the field (based on document coordinates) `required = true`
	posy	integer	y-coordinate of the field (based on document coordinates) required = true
	signingModeOptions	List<SigningMode>	A list of the signing modes which should be allowed for the signer, can be: HW - for handwritten (Tablet or HTML5) PH - photo, captured by camera C2S - 'Click2Sign' Default: HW, PH and C2S `required = false`
RestTextFieldInput	id	string	Id of the field (used as fieldname in PDF document) which must be unique within the document `required = false`
	name	string	The name of the document's field. `required = false`
	signerId	string	The id of the signer (id from RestSignerInput) which is assigned to the field `required = false`
	label	string	The label of the field `required = true`
	required	boolean	Specifies whether the field is mandatory for the signer or not `required = false`
	readOnly	boolean	Specifies whether the field is read only or not `required = false`
	data	string	The value of the text field `required = false`
	height, width, posx and posy	integer	Field coordinates and dimension (see RestSignatureFieldInput) `required = true`

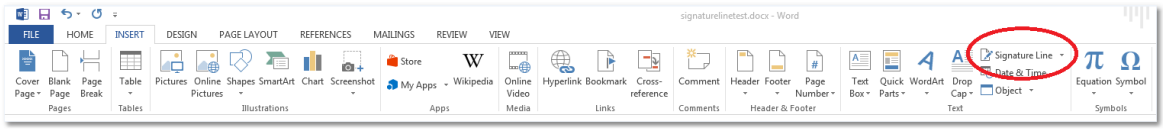
Class	Element	Element Type	Description
	multipleLine	boolean	Specifies whether the field allows multiple lines or not `required = false`
	maxLength	integer	Maximum length of the field (value) `required = false`
RestCheckboxFieldInput	id	string	Id of the field (used as fieldname in PDF document) which must be unique within the document `required = false`
	signerId	string	The id of the signer (id from RestSignerInput) which is assigned to the field `required = false`
	name	string	The name of the document's field. `required = false`
	label	string	The label of the field `required = false`
	required	boolean	Specifies whether the field is mandatory for the signer or not `required = false`
	readOnly	boolean	Specifies whether the field is read only or not `required = false`
	height, width, posx and posy	integer	Field coordinates and dimension (see RestSignatureFieldInput) `required = true`
	state	CheckboxFieldState	The state of the checkbox field, can be CHECKED or UNCHECKED Default: UNCHECKED `required = false`
RestReminderInput	id	string	The id of the reminder. `required = false`
	type	ReminderType	The type of the reminder. Possible values: • (default) AFTER_SEND Signers will be reminded after

 © 2017 Kofax

Class	Element	Element Type	Description
			specified number of days since package's signing started. • BEFORE_EXPIRES Signers will be reminded until specified number of days before the package will be expired. Default: AFTER_SEND `required = false`
	days	short	Number of days, after/before which signers will be reminded. See field type description. `required = true`

2.6.1.6.3 Create Signing Packages with Word

Kofax SignDoc 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.



A separate menu is opened where a User can specify additional information like signer name, email address and instructions. From this menu a user who wants to prepare a document for Kofax SignDoc has several different approaches.

1. Set signer name for a new Kofax SignDoc 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 Scenario 4.

2.6.1.6.3.1 Word Config

Kofax SignDoc differentiates between two ways to map signature line metadata to Kofax SignDoc signer IDs.

The first option is to use the suggested name of signer in the signature line specification.

The second option is to use the Signature Line TagID embedded in the Document itself. For this option the Kofax SignDoc environment has to be configured in the cirrus.properties to specify winword.field.identification.key=field_name. To Access this TagID an application would have to parse the Word Document. Kofax SignDoc will only work with provided TagIDs and will not parse them itself. The following Tables will give an overview of the Metadata provided with a Signature line and the Kofax SignDoc behavior:

winword.field.identification.key=field_name (enabled):

Signature Line	Kofax SignDoc behaviour
tag id	➔ id of RestSigner
Suggested signer	➔ name in RestSigner
Suggested signer's email address	➔ email in RestSigner

winword.field.identification.key=field_name (not enabled):

Signature Line (MS Word)	Kofax SignDoc
tag id	➔ not used
Suggested signer	➔ id of RestSigner
Suggested signer's email address	➔ email in RestSigner

2.6.1.6.3.2 Principle Process

1. The signer object id is created based on the value of the winword.field.identification.key. If the values is **field name**, the signer id becomes the **name of the signature line** which is the same as the signature line TagID. 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 same id, the signature line will be bound to this signer.
   b. If there is no such signer, Kofax SignDoc 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.

 © 2017 Kofax

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.

After processing all signature lines of a document that way, all unassigned signers of the signing package will be removed.

2.6.1.6.4 Scenarios

Scenario 1 - Create Signing Package from Scratch

Scenario 2 - Create Signing Package from Template

Scenario 3 - Create Signing Package from Template with Metadata

Scenario 4 - Create Signing Package with Word Document

2.6.1.6.4.1 Scenario 1 - 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)

```
{
  "name": "package1",
  "type": "PACKAGE",
  "processingType": "SEQ",
  "signers": [
    {
     "order": 0,
     "id": "signer1",
     "email": "signer1@kofax.com",
     "role": "SIGNER",
     "name": "Signer1"
    }
  ],
  "documents": [
    {
     "name":"Rest Test",
   "description":"Rest Test",
   "documentMessage":"Please",
      "fileName":"RestTest1.pdf",
      "signatureFields": [
         {
         "readOnly":"false",
         "required":"false",
         "posy":35,
         "posx":227,
         "width":281,
         "height":92,
         "id":"Applicant",
         "page":1,
         "signerId": "signer1",
         "label":"Applicant"
         }
     ],
     "content":  "base64 encoded PDF"
    }
   ]
}
```

Please 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 parameter as true. It additionally is possible to delete an already existing Singing Package by specifying the delete_existing parameter as true.

2.6.1.6.4.2 Scenario 2 - Create Signing Package from Template

In this scenario we will create a Singing Package from an already existing template. To do that the src_id parameter has to be specified with an existing Template id. For this scenario a Random Template is sufficient. Otherwise use the following sample Template:

Body Parameter (JSON)

```
{
  "name": "template1",
  "type": "TEMPLATE",
  "processingType": "SEQ",
  "signers": [
     {
      "order": 0,
      "id": "signer1",
      "role": "SIGNER",
      "name": "Signer1"
     }
  ],
  "documents": [
     {
      "name":"Rest Test",
   "description":"Rest Test",
   "documentMessage":"Please",
       "fileName":"RestTest1.pdf",
       "signatureFields": [
          {
          "readOnly":"false",
          "required":"false",
          "posy":35,
          "posx":227,
          "width":281,
          "height":92,
          "id":"Applicant",
          "page":1,
          "signerId": "signer1",
          "label":"Applicant"
          }
      ],
      "content":  " base64 encoded PDF "
     }
   ]
}
```

Please 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 Singing Package by specifying the delete_existing parameter as true.

2.6.1.6.4.3 Scenario 3 - Create Signing Package from Template with Metadata

For this scenario we will create a Signing Package from Template with specified metadata in the Body. First we have to create a Template with a document and signer. For that we use the Create Signing Package request and add the following body parameter to the body:

Body Parameter 1 (JSON)

```
{
  "name": "template1",
  "type": "TEMPLATE",
  "processingType": "SEQ",
  "signers": [
     {
      "order": 0,
      "id": "signer1",
      "role": "SIGNER",
      "name": "Signer1"
      }
  ],
  "documents": [
     {
      "name":"Rest Test",
      "description":"Rest Test",
      "documentMessage":"Please",
      "fileName":"RestTest1.pdf",
      "signatureFields": [
         {
          "readOnly":"false",
          "required":"false",
          "posy":35,
          "posx":227,
          "width":281,
          "height":92,
          "id":"Applicant",
          "page":1,
          "signerId": "signer1",
          "label":"Applicant"
          }
       ,
    "content":  " base64 encoded PDF "
     }
  ]
}
```

Please make sure you add a Base64 encoded Document in the Document Content. After the successful creation of the template we use the returned id in our next Create Signing Package request as the src_id parameter. In our last step we specify the following body parameter in the request body:

Body Parameter 2 (JSON)

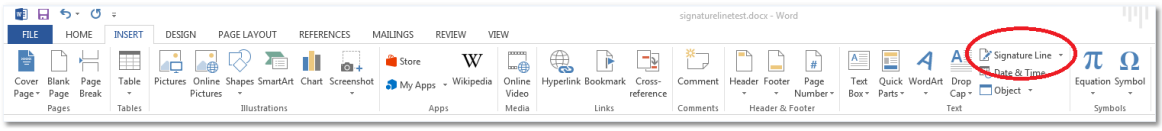
```
{
  "name": "New Package with Signer Email",
  "signers": [
    {
     "id": "signer1",
     "email": "test@test.com"
    }
  ]
}
```

The mapping of Signer email address is accomplished via ID. In the previous Signing Package template we specified the Signer ID as signer1. If we want to set the email address of this Signer we have to specify the predefined ID we already know in the Metadata structure and change the email address to a value of our choosing, see Body Parameter 2 JSON. Additionally we can basically overwrite any other value used to create the Signing Package. As an example we changed the name of the Signing Package to "New Package with Signer Email" instead of "template1". This principal applies to every other properties of the Singing Package a User is allowed to change (see Signing Package Properties Overview chapter).

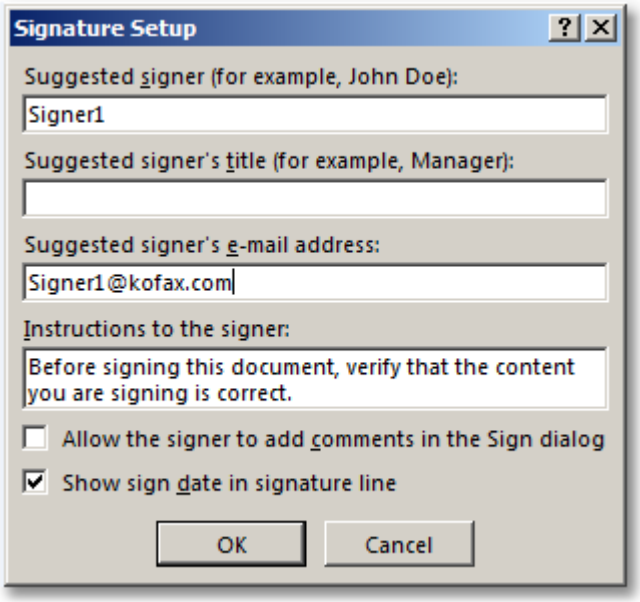
To immediately send the created Signing Package it is only needed to specify the schedule parameter as true. It additionally is possible to delete an already existing Singing Package by specifying the delete_existing parameter as true.

2.6.1.6.4.4 Scenario 4 - Create Signing Package with Word Document

This scenario will use a Word document to create a Signing Package. In the first step the User has to create a Word document with a Signature line. To do that he has to execute the Signature Line feature in the insert submenu on the right side.



In the second step a separate window pops up, to specify additional data for the new Signature Line. The user should specify the suggested signer field (Signer1) and the suggested signer's email address (Signer1@kofax.com).



After pressing the **OK** button a new signature line is added to the Word Document.

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.

 © 2017 Kofax

Body Parameter (JSON)

```
{
  "name": "package1",
  "type": "PACKAGE",
  "processingType": "SEQ",
  "documents": [
     {
      "name":"Rest Test",
      "description":"Rest Test",
      "documentMessage":"Please",
      "format": "MS_WORD",
      "fileName":"RestTest1.docx",
      "content": "base64 encoded word document"
     }
   ]
}
```

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 Singing Package by specifying the delete_existing parameter as true.

2.6.2 Complete Signing Package Structure

RestSigningPackageInput

- id
- name
- type
- auditTrailOptions (default: 3)
- description
- expirationDate
- startDate
- mailMessage and mailSubject
- processingType (default: PAR)
- signingModeOptions
- reminders (a list of RestReminderInput)
  - id
  - days (required)
  - type (default: AFTER_SEND)
- signers (a list of RestSignerInput)
  - id
  - email
  - name
  - order
  - role (default: SIGNER)

    - reasonForDecline
    - commentForDecline
    - authenticationMode (NONE, CODE)
    - authenticationParam
    - accessCodeDeliveryParameters
    - accessCodeDeliveryPluginId
    - clientCertificateRequest (DISABLED, REQUIRED)
    - tspSignerInfo
      - tspCredentials (a list of RestEntry)
        - k
        - v
      - tspPluginId
      - tspSignatureDocuments (a list of IDs of documents)
      - tspSignatureType (BASIC, ADVANCED, QUALIFIED)
- documents (a list of RestDocumentInput)
    - description
    - documentMessage
    - filename
    - name
    - order
    - documentContent
    - content (PDF document base64 encoded)
    - format (PDF)
    - signatureFields (a list of RestSignatureFieldInput)
      - id
      - name
      - readOnly
      - required
      - label
      - signerId
      - page
      - posx
      - posy
      - width
      - height
      - signingModeOptions (default: C2S+HW+PH)
    - textFields (a list of RestTextFieldInput)
      - id

 © 2017 Kofax

- name
- required
- label
- signerId
- readOnly
- page
- posx
- posy
- width
- height
- data
- multipleLine
- maxLength

- checkboxFields (a list of RestCheckboxFieldInput)
  - id
  - name
  - readOnly
  - required
  - label
  - signerId
  - page
  - posx
  - posy
  - width
  - height
  - state (default: UNCHECKED)

2.6.3 Delete a Signing Package

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

NOTE

For this request a valid authentication and User rights are necessary.

URL:	**/rest/v5/packages/{packageid}**
Producible Formats:	JSON, XML
Return Formats:	JSON, XML

Request Header(s):	Accept: application/json, application/xml
HTTP Method:	DELETE

Example Request URL

```
DELETE https://localhost:8080/cirrus/rest/v5/packages/1001
```

Example Headers

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

2.6.3.1 Parameters

Path Parameters

Key	Required	Value Type	Default Value	Description
packageid	yes	string	n/a	ID of the signing package the user want to delete

2.6.4 Update a Signing Package

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

NOTE

For this request a valid authentication and User rights are necessary.

URL:	**/rest/v5/packages/{packageid}**
Producible Formats:	JSON, XML
Return Formats:	JSON, XML
Request Header(s):	Accept: application/json, application/xml
HTTP Method:	PUT

 © 2017 Kofax

Example Request URL

```
PUT https://localhost:8080/cirrus/rest/v5/packages/1001
```

Example Headers

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

Example Request Body (JSON)

```
{
  "id":."package1",
  "name": "package1",
  "type": "PACKAGE",
  "processingType": "SEQ"
}
```

2.6.4.1 Parameters

Path Parameters

Key	Required	Value Type	Default Value	Description
packageid	yes	string	n/a	ID of the signingpackage the user wants to update

Body Parameters

Key	Required	Value Type	Default Value	Description
updPackage	yes	RestSigningPackageInput	null	Represents a RestSigningPackageInput object either in JSON or XML.

2.6.4.2 Response

The response status is 200 if the signing package was successfully updated, otherwise a Kofax SignDoc Status Code is returned together with the explaining message(s).

2.6.4.3 Fields

Available Fields

Key	Value Type	Description
id	string	The id of the package

Key	Value Type	Description
name	string	The name of the package
type	string	The type of the package. This property can be set to either PACKAGE or TEMPLATE.
description	string	The description of the package
processingType	ProcessingType	The processing type of the package. This property can be set to either SEQ or PAR.
expirationDate	date	The expiration date of the package. Format: date-time with a time-zone in UTC, such as '2007-12-03T10:15:30Z' (ISO-8601).
startDate	string	The start date of the package. Format: date-time with a time-zone in UTC, such as '2007-12-03T10:15:30Z' (ISO-8601).
mailSubject	string	The mail subject of the package
mailMessage	string	The mail message of the package
auditTrailOptions	int	The audit trail options of the package
signers	Array[RestSignerInput]	A list of all signers contained in the package
documents	Array[RestDocumentInput]	A list of all documents contained in the package
reminders	Array[RestReminderInput]	A list of all reminders contained in the package
signingModeOptions	Array[String]	A list of the default signing modes. This property can be set to either PH, HW or C2S.

2.6.5 Update an Existing Signature Field with Form Data

This request updates an existing signature field identified by ID with form data. Please note that the page, coordinates and dimensions for the update have to be in bounds of the document.

NOTE

For this request a valid authentication and User rights are necessary.

URL:	**/rest/v5/packages/{packageid}/documents/{documentid}/signaturefield/{signaturefieldid}**
Producible Formats:	JSON, XML

 © 2017 Kofax

Request Header(s):	Accept: application/json, application/xml
HTTP Method:	POST

Example Request URL

```
POST
   https://localhost:8080/cirrus/rest/v5/packages/1001/documents/1002
   /signaturefields/1003
```

Example Headers

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

2.6.5.1 Parameters

Path Parameters

Key	Required	Value Type	Default Value	Description
documentid	yes	integer	n/a	ID of the text field to update
packageid	yes	string	n/a	ID of the signing package to query the audit trail from
signaturefieldid	yes	string	n/a	ID of the signature field to update

Query Parameters

Key	Required	Value Type	Default Value	Description
fieldName	no	string	n/a	New fieldName of signature field
label	no	string	n/a	New label of signature field
required	no	boolean	n/a	New required of signature field
page	no	integer	n/a	New page of signature field
posx	no	integer	n/a	New posx of signature field
posy	no	integer	n/a	New posy of signature field
width	no	integer	n/a	New width of signature field
height	no	integer	n/a	New height of signature field

Key	Required	Value Type	Default Value	Description
signerid	no	integer	n/a	ID of a signer. The specified signer will be assigned as the new signer of the signature field.

2.6.5.2 Response

The response status is 200 if the signature field was successfully updated, otherwise a Kofax SignDoc Status Code is returned together with the explaining message(s).

2.6.6 Get a Single Signing Package

Returns a single signing package specified by a given account ID. Since a signing package usually contains multiple documents and can become quite large, a separate list with RestDocumentListEntry object was introduced to limit the needed size. A RestDocumentListEntry contains only necessary information about a document as well as a separate rest url parameter to query the entire document in a separate request.

NOTE

For this request a valid authentication and User or Signer rights are necessary.

URL:	**/rest/v5/packages/{packageid}**
Producible Formats:	JSON, XML
Request Header(s):	Accept: application/json, application/xml
HTTP Method:	GET

Example Request URL

```
GET https://localhost:8080/cirrus/rest/v5/packages/1001
```

Example Headers

```
Accept: application/json
```

 © 2017 Kofax

Example Response Body (JSON)


```
{
"id": "80ac5d06-281e-4fc2-8ff0-060055304fc5",
"name": "package1",
"type": "PACKAGE",
"state": "STARTED",
"processingType": "SEQ",
"startDate": 1429740000000,
"mailSubject": "An e-signing request via Kofax SignDoc",
"mailMessage": "Admin User has invited you to sign documents with Kofax SignDoc",
"auditTrailOptions": 3,
"lastUpdateTime": 1429792756147,
"creationTime": 1429792756116,
"startTime": 1429792756147,
"auditTrailUrl": "/rest/v5/packages/610/audittrail",
"esignConsent": "Text of e-sign consent"
"owner": {
  "id": "user1",
  "name": "User1",
  "email": "user1@test.com"
  "company": "Company1"
},
"documentEntries": [
  {
    "id": "6b7efc3b-02e8-4ce6-a35b-ab884ae79041",
    "name": "Rest Test",
    "order":0,
    "description": "Rest Test",
    "url": "/rest/v5/packages/80ac5d06-281e-4fc2-8ff0-060055304fc5/documents/6b7efc3b-
02e8-4ce6-a35b-ab884ae79041",
    "fields": []
  }
],
"signerEntries": [
  {
    "id": "signer1",
    "name": "Signer1",
    "email": "signer1@kofax.com",
    "order": 0,
    "role": "SIGNER",
    "state": "ASSIGNED",
    "authenticationMode": "NONE",
    "creationTime": 1429792756116,
    "url": "/rest/v5/packages/80ac5d06-281e-4fc2-8ff0-060055304fc5/signer?
signerid=6b7efc3b-02e8-4ce6-a35b-ab884ae79041"
  }
]
}
```


2.6.6.1 Parameters

Path Parameters

Key	Required	Value Type	Default Value	Description
packageid	yes	string	n/a	ID of the signing package to query

Query Parameters

Key	Required	Value Type	Default Value	Description
useIntId	no	boolean	false	The provided packageid is the internal id which was returned by get audit trail method.

2.6.6.2 Response

The response status is 200 if the package was queried successfully, otherwise a Kofax SignDoc Status Code is returned together with the explaining message(s). In case of a successful query the response body contains the following information as RestSigningPackageOutput structure:

- id
- name
- type
- description
- owner
  - id
  - name
  - company
  - email
- editor
- state (DRAFT, PREPARED, STARTED, COMPLETE, REJECTED, EXPIRED, CANCELED, ARCHIVED)
- esignConsent
- auditTrailOptions
- lastUpdateTime
- creationTime
- timeStarted
- completionTime
- auditTrailUrl
- expirationDate
- startDate
- currentSigner
  - id
  - email
  - name

 © 2017 Kofax

    - order
    - role (SIGNER, REVIEWER)
    - reasonForDecline
    - commentForDecline
    - authenticationMode (NONE, CODE)
    - authenticationTime
    - authenticationParam
    - accessCodeDeliveryChannel
    - accessCodeDeliveryPluginId
    - accessCodeDeliveryParameters
    - completionTime
    - state (ASSIGNED, INFORMED, COMPLETE, REJECTED, ERROR)
    - clientCertificateRequest (DISABLED, REQUIRED)
- mailMessage and mailSubject
- processingType
- reminderEntries (a list of RestReminderOutput)
    - id
    - days
    - type (AFTER_SEND, BEFORE_EXPIRES)
    - sendDate
    - sentTime
- signerEntries (a list of RestSignerListEntry)
    - id
    - email
    - name
    - order
    - role
    - state
    - accessCodeDeliveryChannel
    - clientCertificateRequest (DISABLED, REQUIRED)
    - unreadDocuments
    - tspSignatureDocuments (a list of RestTSPSignatureDocument)
        - documentId
        - state (REQUIRED, DONE)
- documentEntries (a list of RestDocumentListEntry)
    - id
    - description

- documentMessage
- filename
- name
- order
- documentContent
- content
- format

2.6.7 Get Audit Trail

Returns the audit trail for a signing package specified by a given package ID.

NOTE

For this request a valid authentication and USER rights are necessary.

URL:	**/rest/v5/packages/{packageid}/audittrail**
Producible Formats:	JSON, XML
Request Header(s):	Accept: application/json, application/xml
HTTP Method:	GET

Example Request URL

```
GET https://localhost:8080/cirrus/rest/v5/packages/1001/audittrail
```

Example Headers

```
Accept: application/json
```

Example Response Body (JSON)

```
[
 {
    "message": "The signing package Package1 has been created.",
    "workflowEvent": "PKG_CREATED",
    "creationTime": "2017-01-25T11:48:33.466Z",
    "ipAddr": "127.0.0.1",
    "intUserId": 1275,
    "intAccountId": 56,
    "intPackageId": 6051
  }
]
```

 © 2017 Kofax

2.6.7.1 Parameters

Path Parameters

Key	Required	Value Type	Default Value	Description
packageid	yes	string	n/a	ID of the signing package to query the audit trail from

2.6.7.2 Response

The response status is 200 with list of RestAuditTrailOutput entries if the package audit trail was queried successfully, otherwise a Kofax SignDoc Status Code is returned together with the explaining message(s).

Class	Element	Element Type	Description
RestAuditTrailOutput	message	string	The message of the audit trail
	creationTime	string	The creation date of the audit trail. Format: date-time with a time-zone in UTC, such as '2007-12-03T10:15:30Z' (ISO-8601)
	workflowEvent	ERestWorkflowEvent	The workflow event of the audit trail. Possible values: 'PKG_CREATED', 'PKG_STARTED', 'PKG_EXPIRED', 'PKG_COMPLETED', 'PKG_DELETED', 'PKG_REMINDER_ADDED', 'PKG_SIGNING_SESSION_CLOSED', 'SIG_NOTIFIED', '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_AGREE_ESIGN_CONSENT', 'SIG_OPEN_DOCUMENT', 'SIG_MANUALY_AUTHENTICATED', 'SIG_SIGNED', 'SIG_CLEARED_SIGNATURE', 'SIG_CHECKBOX_CHECKED',

Class	Element	Element Type	Description
			'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', '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'
	ipAddr	string	The IP address
	intAccountId	string	The internal ID of the account
	intPackageId	string	The internal ID of the package
	intUserId	string	The internal ID of the user
	intSignerId	string	The internal ID of the signer
	intDocumentId	string	The internal ID of the document
	intDocumentFieldId	string	The internal ID of the document field

 © 2017 Kofax

2.6.8 Get a List of Packages

Returns a list of packages owned by the authenticated user. The resulting list contains only necessary information for further queries, because a complete list of Signing Packages can become very large. For a complete data set please use the Get a Single Signing Package request (rest/package/{id}).

NOTE

For this request a valid authentication and User rights are necessary.

URL:	**/rest/v5/packages**
Producible Formats:	JSON, XML
Request Header(s):	Accept: application/json, application/xml
HTTP Method:	GET

Example Request URL

```
GET https://localhost:8080/cirrus/rest/v5/packages
```

Example Headers

```
Accept: application/json
```

2.6.8.1 Parameters

Query Parameters

Key	Required	Value Type	Default Value	Description
packageTypeFilter	no	SigningPackageType [PACKAGE, TEMPLATE]	n/a	Whether to filter the list for the types SingingPackages or Template or not

2.6.8.2 Response

In case of a successful query the response body contains a list of RestPackageListEntry objects with the following information regarding a single signing packages: id, name, description, owner, type and a rest url to query the entire signing package in a separate request.

RestPackageListEntry

- id
- name
- type
- description
- owner
- editor
- state (DRAFT, PREPARED, STARTED, COMPLETE, REJECTED, EXPIRED, CANCELED, ARCHIVED)
- documentEntries (a list of RestDocumentListEntry)
  - id
  - description
  - name
  - url
  - thumbnail
- signerEntries (a list of RestSignerListEntry)
  - id
  - email
  - name
  - order
  - role
  - state
  - accessCodeDeliveryChannel
  - clientCertificateRequest (DISABLED, REQUIRED)
  - unreadDocuments
  - tspSignatureDocuments (a list of RestTSPSignatureDocument)
    - documentId
    - state (REQUIRED, DONE)

2.6.9 Schedule a Signing Package

This request evaluates the specified signing package and schedules it if it is in a persistent state. Afterwards it triggers the signing package process and sends out the invitations for the signing package. Please note, that the time of the send out invitations depends on the specified start time of the signing package. A package with no start time will start immediately, a package with a start date will be set on hold until the start date has been reached. Please note that a signing package has to be in a consistent state to be scheduled, otherwise persistence errors will occur (e.g. a signing package needs at least one signer). In such cases all persistence errors and detailed information on the problem will be returned in form of a list.

NOTE

For this request a valid authentication and User rights are necessary.

 © 2017 Kofax

NOTE

The request will be denied if there is at least one signer with role SIGNER who does not have any assigned signature field.

URL:	**/rest/v5/packages/{packageid}/scheduler**
Producible Formats:	JSON, XML
Request Header(s):	Accept: application/json, application/xml
HTTP Method:	POST

Example Request URL

```
POST https://localhost:8080/cirrus/rest/v5/packages/1001/scheduler
```

Example Headers

```
Accept: application/json
```

2.6.9.1 Parameters

Path Parameters

Key	Required	Value Type	Default Value	Description
packageid	yes	string	n/a	ID of the signing package

2.6.10 Prepare Signing Session

This request prepares a signing session. In case of signtype common it returns the link and optional the appropriate QR code for starting a 'common' (or 'in-person') signing session. All signers with an email are notified with an invitation for a 'remote' signing session if signtype remote is used.

NOTE

For this request a valid authentication and USER rights are necessary.

URL:	**/rest/v5/packages/{packageid}/signingsession/{signtype}**

Producible Formats:	JSON, XML
Return Formats:	JSON, XML
Request Header(s):	Accept: application/json, application/xml
HTTP Method:	POST

Example Request URL

```
DELETE
   https://localhost:8080/cirrus/rest/v5/packages/1001/signingsession
   /common
```

Example Headers

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

2.6.10.1 Parameters

Path Parameters

Key	Required	Value Type	Default Value	Description
packageid	yes	string	n/a	ID of the signing package the user wants to create a signing session for.
signtype	yes	string	n/a	Signing type of the signing session. Possible values: common, remote

Body Parameters

Key	Required	Value Type	Default Value	Description
commonSigningSessionInput	no	RestCommonSigningSessionInput	null	Represents a RestCommonSigningSessionInput object either in JSON or XML.

The following table specifies the possible properties a user can set in the body parameter.

Class	Element	Element Type	Description
RestCommon	qrCodeSpecificati	RestQRCode	Object that holds specifications for the QR

 © 2017 Kofax

Class	Element	Element Type	Description
SigningSessionInput	ons	Specifications	code.
	manualSignerAuthentications	RestManualSignerAuthentication	A list of objects that hold information about authentication data of a signer.
RestQRCodeSpecifications	imageType	ERestQRCodeImageType	Type of QR code image.
	width	integer	Width of QR code image.
	height	integer	Height of QR code image.
RestManualSignerAuthentication	signerId	string	Id of the signer.
	passport	boolean	Whether the signer was authenticated by passport.
	driverLicense	boolean	Whether the signer was authenticated by driver license.
	visualVerification	boolean	Whether the signer was authenticated by visual verification.
	other	string	Any other means of authentication for the signer.

Example Body (JSON)

```
{
  "manualSignerAuthentications": [
    {
      "signerId": "0f8e861e-032e-435c-bc67-687f337c74ca",
      "passport": true,
      "visualVerification": true,
      "other": "another verification"
    }
  ],
  "qrCodeSpecifications": {
    "imageType": "JPG",
    "width": 200,
    "height": 200
  }
}
```

2.6.10.2 Response

The response status is 200 if the signing session was successfully prepared, otherwise a Kofax SignDoc Status Code is returned together with the explaining message(s).

In case of a successful query the response body contains the following information as RestCommonSigningSessionOutput structure:

Class	Element	Element Type	Description
RestCommonSigningSessionOutput	url	string	Url of the signing session.
	qrcode	string	Base64 encoded qr code of the signing session.

Example Response Body (JSON)

```
{
 "url": "http://ksd.de:80/sdweb/sif/v1/initsign?pid=fce264a&auth=59ce1906e5&signtype=COMMON",
 "qrcode": "Base64EncodedQRCode"
}
```

Example Response Body (XML)

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<RestCommonSigningSessionOutput xmlns="http://www.kofax.com/ksd/cirrus/rest/v5">
 <url>http://ksd.de:80/sdweb/sif/v1/initsign?pid=fce0264a&auth=59ce1906e5&signtype=COMMON</url>
 <qrcode>Base64EncodedQRCode</qrcode>
</RestCommonSigningSessionOutput>
```

2.7 Reminder Requests

Add Reminder to Package

Update a Reminder

Delete a Reminder

2.7.1 Add Reminder to Package

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

NOTE

For this request a valid authentication and User rights are necessary.

URL:	**/rest/v5/packages/{packageid}/reminder**
Producible Formats:	JSON, XML
Return Formats:	JSON, XML
Request Header(s):	Accept: application/json, application/xml

 © 2017 Kofax

HTTP Method:	POST

Example Request URL

```
POST https://localhost:8080/cirrus/rest/v5/packages/1001/reminder
```

Example Headers

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

Example Request Body (JSON)

```
{
   "id" : "reminder1",
   "type": "BEFORE_EXPIRES",
   "days": "3"
}
```

Example Request Body (XML)

```
<RestReminderInput>
  <id>reminder1</>
  <type>BEFORE_EXPIRES</type>
  <days>3</days>
</RestReminderInput>
```

2.7.1.1 Parameters

Path Parameters

Key	Required	Value Type	Default Value	Description
packageid	yes	string	n/a	ID of the signing package the user wants to add a reminder

Body Parameters

Key	Required	Value Type	Default Value	Description
reminder	yes	RestReminderInput	null	Represents a RestReminderInput object either in JSON or XML.

2.7.1.2 Response

The response status is 201 (CREATED) if the reminder was created, otherwise a Kofax SignDoc Status Code is returned together with the explaining message(s).

2.7.1.3 Fields

Mandatory Fields

Key	Value Type	Description
days	short	The days of the reminder

Available Fields

Key	Value Type	Description
id	string	ID of the reminder
type	ReminderType	The type of the reminder. This property can be set to either AFTER_SEND or BEFORE_EXPIRES.
days	short	The days of the reminder

2.7.2 Update a Reminder

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

NOTE

For this request a valid authentication and User rights are necessary.

URL:	**/rest/v5/packages/{packageid}/reminders/{reminderid}**
Producible Formats:	JSON, XML
Return Formats:	JSON, XML
Request Header(s):	Accept: application/json, application/xml
HTTP Method:	PUT

Example Request URL

```
PUT
   https://localhost:8080/cirrus/rest/v5/packages/1001/reminders/1002
```

 © 2017 Kofax

Example Headers

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

Example Request Body (JSON)

```
{
   "id" : "reminder1",
   "type": "BEFORE_EXPIRES",
   "days": "3"
}
```

Example Request Body (XML)

```
<RestReminderInput>
  <id>reminder1</id>
  <type>BEFORE_EXPIRES</type>
  <days>3</days>
</RestReminderInput>
```

2.7.2.1 Parameters

Path Parameters

Key	Required	Value Type	Default Value	Description
packageid	yes	string	n/a	ID of the signing package the user wants to add a reminder
reminderid	yes	string	n/a	ID of the reminder the user wants to update

Body Parameters

Key	Required	Value Type	Default Value	Description
reminder	yes	RestReminderInput	null	Represents a RestReminderInput object either in JSON or XML.

2.7.2.2 Response

The response status is 200 if the reminder was successfully updated, otherwise a Kofax SignDoc Status Code is returned together with the explaining message(s).

2.7.2.3 Fields

Available Fields

Key	Value Type	Description
id	string	The id of the reminder
type	ReminderType	The type of the reminder. This property can be set to either AFTER_SEND or BEFORE_EXPIRES.
days	short	The days of the reminder

2.7.3 Delete a Reminder

Deletes a reminder specified by ID.

NOTE

For this request a valid authentication and User rights are necessary.

URL:	**/rest/v5/packages/{packageid}/reminders/{reminderid}**
Producible Formats:	JSON, XML
Return Formats:	JSON, XML
Request Header(s):	Accept: application/json, application/xml
HTTP Method:	DELETE

Example Request URL

```
DELETE
   https://localhost:8080/cirrus/rest/v5/packages/1001/reminders/1002
```

Example Headers

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

 © 2017 Kofax

2.7.3.1 Parameters

Path Parameters

Key	Required	Value Type	Default Value	Description
packageid	yes	string	n/a	ID of the signing package the user wants to add a reminder
reminderid	yes	string	n/a	ID of the reminder the user wants to delete

2.7.3.2 Response

The response status is 200 if the reminder was successfully deleted, otherwise a Kofax SignDoc Status Code is returned together with the explaining message(s).

2.8 Signer Requests

Add Signer to Package
Get a Single Signer
Delete a Signer
Update a Signer
Create a Signing Authentication Token

2.8.1 Add Signer to Package

Adds a new signer to an existing signing package. An input JSON or XML string specifying the details of the Signer, has to be provided as a request parameter. The specification has to provide at least a signer name with more than two characters or an email address. Please note that a valid signer has to have at least a name, an email address and role (by default SIGNER).

NOTE

For this request a valid authentication and User rights are necessary.

URL:	**/rest/v5/packages/{packageid}/signer**
Producible Formats:	JSON, XML
Return Formats:	JSON, XML

Request Header(s):	Accept: application/json, application/xml
HTTP Method:	POST

Example Request URL

```
POST https://localhost:8080/cirrus/rest/v5/packages/1001/signer
```

Example Headers

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

Example Request Body (JSON)

```
{
  "id" : "signer11",
  "name": "Signer11",
  "email": "signer11@test.com",
  "role": "SIGNER",
  "order": 1
}
```

Example Request Body (XML)

```
<RestSignerInput>
  <id>signer11</id>
  <name>Signer11</name>
  <email>signer11@test.com</email>
  <role>SIGNER</role>
  <order>1</order>
</RestSignerInput>
```

2.8.1.1 Parameters

Path Parameters

Key	Required	Value Type	Default Value	Description
packageid	yes	string	n/a	ID of the signing package the user wants to add a reminder

Body Parameters

Key	Required	Value Type	Default Value	Description
signer	yes	RestSignerInput	null	Represents a RestSignerInput object either in JSON or XML.

 © 2017 Kofax

2.8.1.2 Response

The response status is 201 (CREATED) if the signer was created, otherwise a Kofax SignDoc Status Code is returned together with the explaining message(s).

2.8.1.3 Fields

Mandatory Fields

Key	Value Type	Description
n/a	no	n/a

Available Fields

Key	Value Type	Description
id	string	The id of the signer which must be unique within the package. It is necessary to provide an id for referencing a signer in a document field as signerId. `required = false`
name	string	The name of the signer `required = false`
email	string	The email address of the signer `required = false`
role	SignerRole	Signer role, can be SIGNER or REVIEWER. Default: SIGNER `required = false`
order	short	The order of the signer. Used when package's processingType is SEQ. `required = false`
reasonForDecline	SignerReasonForDecline	The reason for decline of the signer. Possible values: • 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 e-sign consent) `required = false`
commentForDecline	string	The comment for decline of the signer. `required = false`

Key	Value Type	Description
authentication	AuthenticationMode	The authentication mode for the signer. Possible values: • NONE (no additional authentication required) • CODE (access code must be provided by the sender) `required = false`
authenticationParam	string	The authentication code for the signer. `required = false`
authenticationDeliveryChannel	AuthenticationDeliveryChannel	The channel by which authentication code will be sent to the signer. `required = false`
clientCertificateRequest	ERestClientCertificateRequest	The setting specifies whether to use client certificate for signing. It is relevant only when account-specific setting 'client.signing.certificate.request' is set to OPTIONAL. Possible values: • DISABLED • REQUIRED
tspSignerInfo	RestTSPSignerInfoInput	TSP-related info for the signer to sign documents using a Trusted Service Provider (TSP)

2.8.2 Get a Single Signer

Returns a signer specified by its ID and signing package ID.

NOTE

For this request a valid authentication and USER or SIGNER rights are necessary.

URL:	**/rest/v5/packages/{packageid}/signers/{signerid}**
Producible Formats:	JSON, XML
Request Header(s):	Accept: application/json, application/xml
HTTP Method:	GET

 © 2017 Kofax

Example Request URL

```
GET https://localhost:8080/cirrus/rest/v5/packages/1001/signers/1002
```

Example Headers

```
Accept: application/json
```

2.8.2.1 Parameter

Path Parameters

Key	Required	Value Type	Default Value	Description
packageid	yes	string	n/a	Id of the signing package
signerid	no	string	n/a	Id of the signer

Query Parameters

Key	Required	Value Type	Default Value	Description
useIntId	no	boolean	false	The provided ids for package and signer are the internal ids which were returned by get audit trail method.

2.8.2.2 Response

The response status is 200 if the signer was queried successfully, otherwise a Kofax SignDoc Status Code is returned together with the explaining message(s). In case of a successful query the response body contains the following information regarding the requested document as RestSignerOutput structure:

- id
- email
- name
- order
- role (SIGNER, REVIEWER)
- reasonForDecline
- commentForDecline
- authenticationMode (NONE, CODE)

- authenticationTime
- authenticationParam
- accessCodeDeliveryChannel
- accessCodeDeliveryPluginId
- accessCodeDeliveryParameters
- completionTime
- state (ASSIGNED, INFORMED, COMPLETE, REJECTED, ERROR)
- clientCertificateRequest (DISABLED, REQUIRED)
- tspSignerInfo
  - tspCredentials (a list of RestEntry)
    - k
    - v
  - tspPluginId
  - tspSignatureDocuments (a list of RestTSPSignatureDocument)
    - documentId
    - state (REQUIRED, DONE)
  - tspSignatureType (BASIC, ADVANCED, QUALIFIED)

2.8.3 Delete a Signer

Deletes a signer specified by ID.

NOTE

For this request a valid authentication and User rights are necessary.

URL:	**/rest/v5/packages/{packageid}/signers/{signerid}**
Producible Formats:	JSON, XML
Return Formats:	JSON, XML
Request Header(s):	Accept: application/json, application/xml
HTTP Method:	DELETE

Example Request URL

```
DELETE https://localhost:8080/cirrus/rest/v5/packages/signers/1002
```

 © 2017 Kofax

Example Headers

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

2.8.3.1 Parameters

Path Parameters

Key	Required	Value Type	Default Value	Description
packageid	yes	string	n/a	ID of the signing package the user wants to add a reminder
signerid	yes	string	n/a	ID of the signer the user wants to delete

2.8.3.2 Response

The response status is 200 if the signer was successfully deleted, otherwise a Kofax SignDoc Status Code is returned together with the explaining message(s).

2.8.4 Update a Signer

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

NOTE

For this request a valid authentication and User rights are necessary.

URL:	**/rest/v5/packages/{packageid}/signers/{signerid}**
Producible Formats:	JSON, XML
Return Formats:	JSON, XML
Request Header(s):	Accept: application/json, application/xml
HTTP Method:	PUT

Example Request URL

```
PUT https://localhost:8080/cirrus/rest/v5/packages/1001/signers/1002
```

Example Headers

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

Example Request Body (JSON)

```
{
   "id" : "signer11",
   "name": "Signer11",
   "email": "signer11@test.com",
   "role": "SIGNER",
   "order": 1
}
```

Example Request Body (XML)

```
<RestSignerInput>
  <id>signer11</id>
  <name>Signer11</name>
  <email>signer11@test.com</email>
  <role>SIGNER</role>
  <order>1</order>
</RestSignerInput>
```

2.8.4.1 Parameters

Path Parameters

Key	Required	Value Type	Default Value	Description
packageid	yes	string	n/a	ID of the signing package the user wants to add a reminder
signerid	yes	string	n/a	The ID of the signer the user wants to update

Body Parameters

Key	Required	Value Type	Default Value	Description
signer	yes	RestSignerInput	null	Represents a RestSignerInput object either in JSON or XML.

2.8.4.2 Response

The response status is 200 if the signer was successfully updated, otherwise a Kofax SignDoc Status Code is returned together with the explaining message(s).

 © 2017 Kofax

2.8.4.3 Fields

Mandatory Fields

Key	Value Type	Description
n/a	no	n/a

Available Fields

Key	Value Type	Description
id	string	The id of the signer which must be unique within the package. It is necessary to provide an id for referencing a signer in a document field as signerId. `required = false`
name	string	The name of the signer `required = false`
email	string	The email address of the signer `required = false`
role	SignerRole	Signer role, can be SIGNER or REVIEWER. Default: SIGNER `required = false`
order	short	The order of the signer. Used when package's processingType is SEQ. `required = false`
reasonForDecline	SignerReasonForDecline	The reason for decline of the signer. Possible values: • 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 e-sign consent) `required = false`
commentForDecline	string	The comment for decline of the signer. `required = false`
authentication	AuthenticationMode	The authentication mode for the signer. Possible values: • NONE (no additional authentication required) • CODE (access code must be provided by the sender)

Key	Value Type	Description
		`required = false`
authenticationParam	string	The authentication code for the signer. `required = false`
authenticationDeliveryChannel	AuthenticationDeliveryChannel	The channel by which authentication code will be sent to the signer. `required = false`
clientCertificateRequest	ERestClientCertificateRequest	The setting specifies whether to use client certificate for signing. It is relevant only when account-specific setting 'client.signing.certificate.request' is set to OPTIONAL. Possible values: • DISABLED • REQUIRED.

2.8.5 Create a Signing Authentication Token

Generates a timely 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 separate by a dot. E.g. BASE64(RestSigningAuthentication) + '.' + BASE64 (hash value of RestSigningAuthentication).

The RestSigningAuthentication object can be used to access important information like the accountId, the packageId or the expiry date of the authentication token. The generated signing authentication token is returned as X-S-AUTH-TOKEN response header.

To authenticate via the generated authentication token a signer has to send the authentication token as part of the X-S-AUTH-TOKEN header for each new request. The content type of the body is application/x-www-form-urlencoded.

URL:	**/rest/v5/signers/authentication**
Producible Formats:	JSON, XML
Return Formats:	JSON, XML
Request Header(s):	Accept: application/json, application/xml
HTTP Method:	POST

Example Request URL

```
POST https://localhost:8080/cirrus/rest/v5/signers/authentication
```

 © 2017 Kofax

Example Headers

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

2.8.5.1 Parameters

Query Parameters

Key	Required	Value Type	Default Value	Description
token	yes	string	n/a	The signing session token
signtype	yes	string	n/a	The signing type (‘REMOTE’ or ‘COMMON’)
signerid	no	string	n/a	The id of signer. It should be provided in case of Common Signing Session for document processing requests.
accesscode	no	string	n/a	The access code requested for two-factor authentication of a remote signer

2.8.5.2 Response

The response status is 200 if the signing authentication token was successfully created, otherwise a Kofax SignDoc Status Code is returned together with the explaining message(s).

2.9 System Requests

Get Timezones
Get the Version of Kofax SignDoc Distribution

2.9.1 Get Time Zones

Returns all available time zones.

NOTE

For this request a valid authentication and Account Administrator or Super User rights are necessary.

URL:	**/rest/v5/system/timezones**
Producible Formats:	JSON, XML
Request Header(s):	Accept: application/json, application/xml
HTTP Method:	GET

Example Request URL

```
GET https://localhost:8080/cirrus/rest/v5/system/timezones
```

Example Headers

```
Accept: application/json
```

Example Response Body (JSON)

```
[
  {
    "timeZoneId": "Etc/GMT+12",
    "displayName": "(GMT -12:00) International Date Line West"
  },
  {
    "timeZoneId": "Etc/GMT+11",
    "displayName": "(GMT -11:00) Coordinated Universal Time-11"
  },
  {
    "timeZoneId": "Pacific/Honolulu",
    "displayName": "(GMT -10:00) Hawaii"
  },
  …
  {
     "timeZoneId": "Pacific/Apia",
     "displayName": "(GMT +13:00) Samoa"
  }
]
```

2.9.1.1 Response

The response status is 200 if the time zones were queried successfully, otherwise a Kofax SignDoc Status Code is returned together with the explaining message(s).

2.9.2 Get the Version of Kofax SignDoc Distribution

Returns the versions of the of Kofax SignDoc distribution. For this request no authentication necessary.

Returns the versions of the used modules.

NOTE

For this request a valid authentication and User rights are necessary.

 © 2017 Kofax

URL:	**/rest/v5/system/version**
Producible Formats:	JSON, XML
Request Header(s):	Accept: application/json, application/xml
HTTP Method:	GET

Example Request URL

```
GET https://localhost:8080/cirrus/rest/v5/system/version
```

Example Headers

```
Accept: application/json
```

2.9.2.1 Response

The response status is 200 if the versions were queried successfully, otherwise a Kofax SignDoc Status Code is returned together with the explaining message(s).

2.10 Team Requests

Add Team to Account
Update a Team
Delete Team from Account
Remove User from Team
Add User to Team

2.10.1 Add Team to Account

Adds a team to Account. An input JSON or XML string specifying the details of the team, has to be provided as a request parameter. The specification has to provide at least a team name with more than two characters.

NOTE

For this request a valid authentication and Account Administrator rights are necessary.

URL:	**/rest/v5/team**
Producible Formats:	JSON, XML

Return Formats:	JSON, XML
Request Header(s):	Accept: application/json, application/xml
HTTP Method:	POST

Example Request URL

```
POST https://localhost:8080/cirrus/rest/v5/team
```

Example Headers

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

Example Request Body (JSON)

```
{
  "name": " Team1 "
}
```

2.10.1.1 Parameters

Body Parameters

Key	Required	Value Type	Default Value	Description
team	yes	RestTeamInput	null	Input JSON or XML string team

2.10.1.2 Fields

Mandatory Fields

Key	Value Type	Description
name	string	Name of user

Available Fields

Key	Value Type	Description
id	string	The id of the team
name	string	The name of the team
managers	string	A list of all team managers specified by id or email

 © 2017 Kofax

Key	Value Type	Description
members	string	A list of all team members specified by id or email

2.10.1.3 Response

The response status is 201 (CREATED) if the team was created, otherwise a Kofax SignDoc Status Code is returned together with the explaining message(s).

2.10.2 Update a Team

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

NOTE

For this request a valid authentication and Account Administrator or Team Manager rights are necessary.

URL:	**/rest/v5/teams/{teamid}**
Producible Formats:	JSON, XML
Return Formats:	JSON, XML
Request Header(s):	Accept: application/json, application/xml
HTTP Method:	PUT

Example Request URL

```
PUT https://localhost:8080/cirrus/rest/v5/teams/team1
```

Example Headers

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

Example Request Body (JSON)

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

2.10.2.1 Parameters

Path Parameters

Key	Required	Value Type	Default Value	Description
teamid	yes	integer	n/a	ID of the team the user wants to update

Body Parameters

Key	Required	Value Type	Default Value	Description
team	yes	RestTeamInput	null	Represents a RestTeam object either in JSON or XML.

2.10.2.2 Fields

Available Fields

Key	Value Type	Description
id	string	The id of the team
name	string	The name of the team
managers	string	A list of all team managers specified by id or email
members	string	A list of all team members specified by id or email

2.10.2.3 Response

The response status is 200 if the team was successfully updated, otherwise a Kofax SignDoc Status Code is returned together with the explaining message(s).

2.10.3 Delete Team from Account

This request deletes an existing team from the account.

NOTE

For this request a valid authentication and Account Administrator or Team Manager rights are necessary.

URL:	**/rest/v5/teams/{teamid}**

 © 2017 Kofax

Producible Formats:	JSON, XML
Return Formats:	JSON, XML
Request Header(s):	Accept: application/json, application/xml
HTTP Method:	DELETE

Example Request URL

```
DELETE https://localhost:8080/cirrus/rest/v5/teams/team1
```

Example Headers

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

2.10.3.1 Parameters

Path Parameters

Key	Required	Value Type	Default Value	Description
teamid	yes	integer	n/a	ID of the team the user wants to delete

2.10.3.2 Response

The response status is 200 if the Team was successfully deleted, otherwise a Kofax SignDoc Status Code is returned together with the explaining message(s).

2.10.4 Remove User from Team

Removes an existing user from a team. Please note, that the actual user is not deleted. Only the relation between specified user and team is deleted.

NOTE

For this request a valid authentication and Account Administrator or Team Manager rights are necessary.

URL:	**/rest/v5/teams/{teamid}/users/{userid}**
Producible Formats:	JSON, XML

Return Formats:	JSON, XML
Request Header(s):	Accept: application/json, application/xml
HTTP Method:	DELETE

Example Request URL

```
DELETE https://localhost:8080/cirrus/rest/v5/teams/team1/users/user1
```

Example Headers

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

2.10.4.1 Parameters

Path Parameters

Key	Required	Value Type	Default Value	Description
teamid	yes	integer	n/a	ID of the team
userid	yes	integer	n/a	ID of the user who should be deleted

Query Parameters

Key	Required	Value Type	Default Value	Description
manager	no	boolean	false	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.

2.10.4.2 Response

The response status is 200 if the user was successfully removed from the team, otherwise a Kofax SignDoc Status Code is returned together with the explaining message(s).

 © 2017 Kofax

2.10.5 Add User to Team

Adds an existing user to a team. It is possible to declare the added user as the team manager of the team, by setting the asManager property.

NOTE

For this request a valid authentication and Account Administrator or Team Manager rights are necessary.

URL:	**/rest/v5/teams/{teamid}/users/{userid}**
Producible Formats:	JSON, XML
Return Formats:	JSON, XML
Request Header(s):	Accept: application/json, application/xml
HTTP Method:	POST

Example Request URL

```
POST https://localhost:8080/cirrus/rest/v5/teams/team1/users/user1
```

Example Headers

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

2.10.5.1 Parameters

Path Parameters

Key	Required	Value Type	Default Value	Description
teamid	yes	integer	n/a	ID of the team
userid	yes	integer	n/a	ID of the user who should be added

Query Parameters

Key	Required	Value Type	Default Value	Description
as_team_manager	no	boolean	false	Whether the added user should have the team manager or not

2.10.5.2 Response

The response status is 200 if the user was successfully added, otherwise a Kofax SignDoc Status Code is returned together with the explaining message(s).

2.11 User Requests

Create an Authentication Token
Get Number of Users
Create 'password recovery' Notification for a User
Refresh a new Authentication Token
Get Current Server Administrator
Create Server Administrator
Get all Server Administrators
Delete a Server Administrator
Get a Single Server Administrator
Update a Server Administrator
Invite or Reinvite a Server Administrator
Reset Password of a Server Administrator
Add User to Account
Get Current User
Get all Users
Get a Single User
Update a User
Delete a User
Invite or Reinvite a User
Reset Password
Add Role to User
Remove Role from User

2.11.1 Create an Authentication Token

This request creates a timely 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 separate by a dot. E.g. 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.

 © 2017 Kofax

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:	**/rest/v5/users/authentication**
Producible Formats:	JSON, XML
Return Formats:	JSON, XML
Request Header(s):	Accept: application/json, application/xml
HTTP Method:	POST

Example Request URL

```
POST https://localhost:8080/cirrus/rest/v5/users/authentication
```

Example Headers

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

2.11.1.1 Parameters

Query Parameters

Key	Required	Value Type	Default Value	Description
accountid	no	string	n/a	The account ID. This parameter can be omitted if the user is not bound to an account. This is the case for users with the role SUPERUSER.
credentials	yes	string	n/a	Either the userid or email address of the user
password	yes	string	n/a	The password of the user
newpassword	no	string	n/a	This field can be set to specify a new password for the user.

2.11.1.2 Response

The response status is 201 if the authentication token was successfully created, otherwise a Kofax SignDoc Status Code is returned together with the explaining message(s).

2.11.2 Get Number of Users

Returns the number of users inside an account.

NOTE

For this request a valid authentication and Account Administrator or Super User rights are necessary.

URL:	**/rest/v5/users/count**
Producible Formats:	JSON, XML
Return Formats:	JSON, XML
Request Header(s):	Accept: application/json, application/xml
HTTP Method:	GET

Example Request URL

```
GET https://localhost:8080/cirrus/rest/v5/users/count
```

Example Headers

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

Example Response Body (JSON)

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

2.11.2.1 Parameters

Query Parameters

Key	Required	Value Type	Default Value	Description
accountid	no	string	n/a	The account ID. This parameter can be omitted if the user is not bound to an account. This is the case for users with the role SUPERUSER.

 © 2017 Kofax

2.11.2.2 Response

The response status is 200 if the RestCount object was queried successfully, otherwise a Kofax SignDoc Status Code is returned together with the explaining message(s).

2.11.3 Create 'password recovery' Notification for a User

This request is used, when a user has forgotten his password. An email notification is send to the user enabling him to set a new password.

NOTE

For this request no authentication is necessary.

URL:	**/rest/v5/users/{userid}/pwrecovery**
Producible Formats:	JSON, XML
Return Formats:	JSON, XML
Request Header(s):	Accept: application/json, application/xml
HTTP Method:	POST

Example Request URL

```
POST https://localhost:8080/cirrus/rest/v5/users/user1/pwrecovery
```

Example Headers

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

2.11.3.1 Parameters

Path Parameters

Key	Required	Value Type	Default Value	Description
userid	yes	string	n/a	Either the userid or email address of the user

Query Parameters

Key	Required	Value Type	Default Value	Description
accountid	no	string	n/a	This parameter can be omitted if the user is bound to an account.

2.11.3.2 Response

The response status is 200 if the recovery notification was send successfully or the notification was not sent because of ID or e-mail of a user is not correct, otherwise a Kofax SignDoc Status Code is returned together with the explaining message(s).

2.11.4 Refresh a new 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:	**/rest/v5/users/refreshtoken**
Producible Formats:	JSON, XML
Return Formats:	JSON, XML
Request Header(s):	Accept: application/json, application/xml
HTTP Method:	GET

Example Request URL

```
GET https://localhost:8080/cirrus/rest/v5/users/refreshtoken
```

Example Headers

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

2.11.4.1 Response

The response status is 200 if the authentication token was successfully refreshed, otherwise a Kofax SignDoc Status Code is returned together with the explaining message(s).

 © 2017 Kofax

2.11.5 Get Current Server Administrator

Returns the current Server Administrator.

NOTE

For this request a valid authentication and Super User rights are necessary.

URL:	**/rest/v5/users/serveradmin**
Producible Formats:	JSON, XML
Return Formats:	JSON, XML
Request Header(s):	Accept: application/json, application/xml
HTTP Method:	GET

Example Request URL

```
GET https://localhost:8080/cirrus/rest/v5/users/serveradmin
```

Example Headers

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

Example Response Body (JSON)

```
{
  "id": "ksdadmin",
  "name": "Tenants Administrator",
  "email": "ksdadmin@kofax.com",
  "state": "ACTIVE",
  "settings": {
    "defaultTemplateName": "Template created by Tenants Administrator at 11/17/2015 11:17
AM",
    "defaultPackageName": "Package created by Tenants Administrator at 11/17/2015 11:17
AM",
    "defaultConfirmationMessage": "I agree doing e-signing as a legal signing process with
Tenants Administrator",
    "defaultMailSubject": "An e-signing request via Kofax SignDoc",
    "defaultMailMessage": "Tenants Administrator has invited you to sign documents with
Kofax SignDoc"
  },
  "roles": [
    "SUPERUSER"
  ],
  "creationTime": 1444743172375,
  "url": "/rest/v5/users/serveradmins/ksdadmin",
  "timeZone": {
    "timeZoneId": "Europe/Berlin",
    "displayName": "(GMT +01:00) Amsterdam, Berlin, Bern, Rome, Stockholm, Vienna"
  }
}
```

2.11.5.1 Response

The response status is 200 if the Server Administrator was successfully queried, otherwise a Kofax SignDoc Status Code is returned together with the explaining message(s).

2.11.6 Create Server Administrator

To create a Server Administrator, an input JSON or XML string specifying the details of the user, has to be provided as a request parameter. The specification has to provide at least a user name with more than two characters and an email address. To specify an initial state the following choices are available: ACTIVE, SUSPENDED and INVITED. To specify the state ACTIVE or INVITED the user password has to be specified as well, because the user will be activated immediately. If no state is specified ACTIVE will be set as default.

NOTE

For this request a valid authentication and Super User rights are necessary.

URL:	**/rest/v5/users/serveradmin**
Producible Formats:	JSON, XML
Return Formats:	JSON, XML
Request Header(s):	Accept: application/json, application/xml
HTTP Method:	POST

Example Request URL

```
POST https://localhost:8080/cirrus/rest/v5/users/serveradmin
```

Example Headers

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

Example Request Body (JSON)

```
{
  "id": "user1",
  "name": "user1",
  "password": "Test123!",
  "email": "user1@ksd.com",
  "roles": ["SUPERUSER"]
}
```

 © 2017 Kofax

2.11.6.1 Parameters

Body Parameters

Key	Required	Value Type	Default Value	Description
account	yes	RestUserInput	n/a	Input JSON or XML string of user

2.11.6.2 Fields

Mandatory Fields

Key	Value Type	Description
name	string	Name of user
email	string	Email of user
password	string	For the user state ACTIVE a password has to be specified

Available Fields

Key	Value Type	Description
id	string	ID of the user
name	string	Name of the user
email	string	Email of the user
phone	string	Phone of the user
state	UserState	State of the user
settings	string	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	UserRoleName	A list of all roles the user possesses. Possible values: USER, TEAMMGR, ADMIN, SUPERUSER
password	string	Password of the user
timeZone	TimeZone	Time zone of the user

2.11.6.3 Response

The response status is 201 if the Server Administrator was successfully created, otherwise a Kofax SignDoc Status Code is returned together with the explaining message(s).

2.11.7 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 please use the Get a Single Server Administrator request (/rest/account/{id}).

NOTE

For this request a valid session and Super User rights are necessary.

URL:	**/rest/v5/users/serveradmins**
Producible Formats:	JSON, XML
Return Formats:	JSON, XML
Request Header(s):	Accept: application/json, application/xml
HTTP Method:	GET

Example Request URL

```
GET https://localhost:8080/cirrus/rest/v5/users/serveradmins
```

Example Headers

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

Example Response Body (JSON)

```
[
  {
    "id": "ksdadmin",
    "name": "Tenants Administrator",
    "email": "ksdadmin@kofax.com",
    "state": "ACTIVE",
    "url": "/rest/v5/users/serveradmins/ksdadmin",
    "roles": [
      "SUPERUSER"
    ]
  },
  {
    "id": "user1",
    "name": "user1",
    "email": "user1@ksd.com",
    "state": "ACTIVE",
    "url": "/rest/v5/users/serveradmins/user1",
    "roles": [
      "SUPERUSER"
    ]
  }
]
```

 © 2017 Kofax

2.11.7.1 Response

The response status is 200 if the Server Administrators were queried successfully, otherwise a Kofax SignDoc Status Code is returned together with the explaining message(s).

2.11.8 Delete a Server Administrator

Deletes an existing Server Administrator specified by user ID. The currently authenticated user cannot delete itself.

NOTE

For this request a valid authentication and Super User rights are necessary.

URL:	**/rest/v5/users/serveradmins/{userid}**
Producible Formats:	JSON, XML
Return Formats:	JSON, XML
Request Header(s):	Accept: application/json, application/xml
HTTP Method:	DELETE

Example Request URL

```
DELETE
   https://localhost:8080/cirrus/rest/v5/users/serveradmins/ksdadmin
```

Example Headers

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

2.11.8.1 Parameters

Path Parameters

Key	Required	Value Type	Default Value	Description
userid	yes	string	n/a	ID of the Server Administrator who should be deleted

2.11.8.2 Response

The response status is 200 if the Server Administrator was successfully deleted, otherwise a Kofax SignDoc Status Code is returned together with the explaining message(s).

2.11.9 Get a Single Server Administrator

Returns a single Server Administrator specified by a given ID.

NOTE

For this request a valid authentication and Account Administrator or Super User rights are necessary.

URL:	**/rest/v5/ users/serveradmins/{userid}**
Producible Formats:	JSON, XML
Request Header(s):	Accept: application/json, application/xml
HTTP Method:	GET

Example Request URL

```
GET https://localhost:8080/cirrus/rest/v5/users/serveradmins/ksdadmin
```

Example Headers

```
Accept: application/json
```

Example Response Body (JSON)

```
{
  "id": "ksdadmin",
  "name": "Tenants Administrator",
  "email": "ksdadmin@kofax.com",
  "state": "ACTIVE",
  "roles": [
    "SUPERUSER"
  ],
  "creationTime": 1448268442365,
  "url": "/rest/v5/users/serveradmins/ksdadmin",
  "timeZone": {
    "timeZoneId": "Etc/GMT",
    "displayName": "(GMT) Coordinated Universal Time"
  }
}
```

 © 2017 Kofax

2.11.9.1 Parameters

Path Parameters

Key	Required	Value Type	Default Value	Description
userid	yes	string	n/a	ID of user

Query Parameters

Key	Required	Value Type	Default Value	Description
useIntId	no	boolean	false	The provided userid is the internal id which was returned by get audit trail method.

2.11.9.2 Response

The response status is 200 if the Server Administrator was queried successfully, otherwise a Kofax SignDoc Status Code is returned together with the explaining message(s).

2.11.10 Update a Server Administrator

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

NOTE

For this request a valid authentication and Super User rights are necessary.

URL:	**/rest/v5/users/serveradmins/{userid}**
Producible Formats:	JSON, XML
Return Formats:	JSON, XML
Request Header(s):	Accept: application/json, application/xml
HTTP Method:	PUT

Example Request URL

```
PUT https://localhost:8080/cirrus/rest/v5/users/serveradmins/ksdadmin
```

Example Headers

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

Example Request Body (JSON)

```
{
  "password": "Test123!",
  "email": " ksdadmin@ksd.com ",
  "name": "ksdadmin"
}
```

2.11.10.1 Parameters

Path Parameters

Key	Required	Value Type	Default Value	Description
userid	yes	integer	n/a	ID of the user to update

Body Parameters

Key	Required	Value Type	Default Value	Description
user	yes	RestUserInput	n/a	Input JSON or XML string of user

2.11.10.2 Fields

Available Fields

Key	Value Type	Description
id	string	The id of the user
name	string	The name of the user
email	string	The email of the user
phone	string	The phone of the user
state	UserState	The state of the user
settings	string	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

 © 2017 Kofax

Key	Value Type	Description
		$USER which is replaced by the current user name and $NOW which is replaced by the current time.
roles	UserRoleName	A list of all roles the user possesses: Possible values: USER, TEAMMGR, ADMIN, SUPERUSER
password	string	The password of the user
timeZone	TimeZone	The time zone of the user

2.11.10.3 Response

The response status is 200 if the Server Administrator was successfully updated, otherwise a Kofax SignDoc Status Code is returned together with the explaining message(s).

2.11.11 Invite or Reinvite a Server Administrator

This request sends an invitation or a reinvitation to a Server Administrator specified by ID.

NOTE

For this request a valid authentication and Super User rights are necessary.

URL:	**/rest/v5/users/serveradmins/{userid}/invitation**
Producible Formats:	JSON, XML
Return Formats:	JSON, XML
Request Header(s):	Accept: application/json, application/xml
HTTP Method:	POST

Example Request URL

```
POST
   https://localhost:8080/cirrus/rest/v5/users/serveradmins/admin1/in
   vitation
```

Example Headers

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

2.11.11.1 Parameters

Path Parameters

Key	Required	Value Type	Default Value	Description
userid	yes	string	n/a	ID of user

Body Parameters

Key	Required	Value Type	Default Value	Description
user	no	n/a	n/a	n/a

2.11.11.2 Response

The response status is 200 if the Server Administrator was successfully invited or reinvited, otherwise a Kofax SignDoc Status Code is returned together with the explaining message(s).

2.11.12 Reset Password of a Server Administrator

This requests resets the password of a Server Administrator identified by the ID or email address (@ is encoded as %40).

NOTE

For this request a valid authentication and Super User rights are necessary.

URL:	**/rest/v5/users/serveradmins/{userid}/pwreset**
Producible Formats:	JSON, XML
Return Formats:	JSON, XML
Request Header(s):	Accept: application/json, application/xml
HTTP Method:	POST

Example Request URL

```
POST
   https://localhost:8080/cirrus/rest/v5/users/serveradmins/kssdadmin
   /pwreset
```

 © 2017 Kofax

Example Headers

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

2.11.12.1 Parameters

Path Parameter

Key	Required	Value Type	Default Value	Description
userid	yes	string	n/a	ID of the Server Administrator

2.11.12.2 Response

The response status is 200 if the password was successfully reseted, otherwise a Kofax SignDoc Status Code is returned together with the explaining message(s).

2.11.13 Add User to Account

To add a user to an existing Account specified by ID, an input JSON or XML string specifying the details of the user, has to be provided as a request parameter. The specification has to provide at least a user name with more than two characters and an email address. To specify an initial state the following choices are available: ACTIVE, SUSPENDED and INVITED. To specify the state ACTIVE or INVITED the user password has to be specified as well, because the user will be activated immediately. If no state is specified ACTIVE will be set as default.

NOTE

For this request a valid authentication and Account Administrator rights are necessary.

URL:	**/rest/v5/user**
Producible Formats:	JSON, XML
Return Formats:	JSON, XML
Request Header(s):	Accept: application/json, application/xml
HTTP Method:	POST

Example Request URL

```
POST https://localhost:8080/cirrus/rest/v5/user
```

Example Headers

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

Example Request Body (JSON)

```
{
  "passWord": " doe ",
  "email": " john.doe@kofax.com ",
  "userName": " john "
}{
  "id": "user1",
  "name": "user1",
  "email": "user1@ksd.com",
  "state": "ACTIVE",
  "roles": [
    "USER",
    "TEAMMGR",
    "ADMIN"
  ],
  "email": " john.doe@ksd.com ",
  "userName": " john "
}
```

2.11.13.1 Parameters

Query Parameters

Key	Required	Value Type	Default Value	Description
accountid	no	string	n/a	The account ID. This parameter can be omitted if the user is not bound to an account. This is the case for users with the role SUPERUSER.

Body Parameters

Key	Required	Value Type	Default Value	Description
user	yes	RestUserInput	n/a	Represents a RestUserInput object either in JSON or XML.

2.11.13.2 Fields

Mandatory Fields

Key	Value Type	Description
name	string	Name of user
email	string	Email of user

 © 2017 Kofax

Key	Value Type	Description
password	string	For the user state ACTIVE a password has to be specified

Available Fields

Key	Value Type	Description
id	string	ID of the user
name	string	Name of the user
email	string	Email of the user
phone	string	Phone of the user
state	UserState	State of the user
settings	string	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	UserRoleName	List of all roles the user possesses: Possible values: USER, TEAMMGR, ADMIN, SUPERUSER
password	string	Password of the user
timeZone	TimeZone	Time zone of the user

2.11.13.3 Response

The response status is 201 if the user was successfully added, otherwise a <%CIRRUS%> Status Code is returned together with the explaining message(s).

2.11.14 Get Current User

Returns the current user specified by the current authentication.

NOTE

For this request a valid authentication and User rights are necessary.

URL:	**/rest/v5/user**
Producible Formats:	JSON, XML
Return Formats:	JSON, XML

Request Header(s):	Accept: application/json, application/xml
HTTP Method:	GET

Example Request URL

```
GET https://localhost:8080/cirrus/rest/v5/user
```

Example Headers

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

2.11.14.1 Response

The response status is 200 if the user was successfully queried, otherwise a Kofax SignDoc Status Code is returned together with the explaining message(s).

2.11.15 Get all Users

This request returns all users of an account. The resulting list contains only necessary information for further queries sorted by last update time descending. For a complete data set please use the Get a Single User request (/rest/users/{userid}).

NOTE

For this request a valid session and Account Administrator or Super User rights are necessary.

URL:	**/rest/v5/users**
Producible Formats:	JSON, XML
Return Formats:	JSON, XML
Request Header(s):	Accept: application/json, application/xml
HTTP Method:	GET

Example Request URL

```
GET https://localhost:8080/cirrus/rest/v5/users
```

Example Headers

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

 © 2017 Kofax

Example Response Body (JSON)


```
[
  {
    "id": "user1",
    "name": "user1",
    "email": "user1@ksd.com",
    "state": "ACTIVE",
    "url": "/rest/v5/users/user1",
    "roles": [
      "USER",
      "TEAMMGR",
      "ADMIN"
    ]
  },
  {
    "id": "user2",
    "name": "user2",
    "email": "user2@ksd.com",
    "state": "ACTIVE",
    "url": "/rest/v5/users/user2",
    "roles": [
      "USER",
      "TEAMMGR"
    ]
  },
  {
    "id": "user3",
    "name": "user3",
    "email": "user3@ksd.com",
    "state": "ACTIVE",
    "url": "/rest/v5/users/user3",
    "roles": [
      "USER",
      "TEAMMGR",
      "ADMIN"
    ]
  }
]
```


2.11.15.1 Parameters

Query Parameters

Key	Required	Value Type	Default Value	Description
accountid	no	string	n/a	This parameter can be omitted if the user is bound to an account.

2.11.15.2 Response

The response status is 200 if the users were queried successfully, otherwise a Kofax SignDoc Status Code is returned together with the explaining message(s).

2.11.16 Get a Single User

Returns a single user specified by a given user and account ID.

NOTE

For this request a valid authentication and Account Administrator or Super User rights are necessary.

URL:	**/rest/v5/users/{userid}**
Producible Formats:	JSON, XML
Request Header(s):	Accept: application/json, application/xml
HTTP Method:	GET

Example Request URL

```
GET https://localhost:8080/cirrus/rest/v5/users/user1
```

Example Headers

```
Accept: application/json
```

Example Response Body (JSON)

```
{
  "id": "user1",
  "name": "user1",
  "email": "user1@ksd.com",
  "state": "ACTIVE",
  "roles": [
    "USER",
    "TEAMMGR"
  ],
  "lastSignInTime": 1447315855466,
  "creationTime": 1445853879453,
  "url": "/rest/v5/users/user1",
  "timeZone": {
    "timeZoneId": "Europe/Berlin",
    "displayName": "(GMT +01:00) Amsterdam, Berlin, Bern, Rome, Stockholm, Vienna"
  }
}
```

2.11.16.1 Parameters

Path Parameters

Key	Required	Value Type	Default Value	Description
userid	yes	integer	n/a	ID of the user to query

 © 2017 Kofax

Query Parameters

Key	Required	Value Type	Default Value	Description
accountid	no	string	n/a	The account ID. This parameter can be omitted if the user is not bound to an account. This is the case for users with the role SUPERUSER.
useIntId	no	boolean	false	The provided userid is the internal id which was returned by get audit trail method.

2.11.16.2 Response

The response status is 200 if the user was queried successfully, otherwise a Kofax SignDoc Code is returned together with the explaining message(s).

2.11.17 Update a User

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

NOTE

For this request a valid authentication and Account Administrator or Super User rights are necessary.

URL:	**/rest/v5/users/{userid}**
Producible Formats:	JSON, XML
Return Formats	JSON, XML
Request Header(s):	Accept: application/json, application/xml
HTTP Method:	PUT

Example Request URL

```
PUT https://localhost:8080/cirrus/rest/v5/users/user1
```

Example Headers

```
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"
  ],
  "email": " john.doe@ksd.com ",
  "userName": " john "
}
```

2.11.17.1 Parameters

Path Parameters

Key	Required	Value Type	Default Value	Description
userid	yes	integer	n/a	ID of the user to update

Query Parameters

Key	Required	Value Type	Default Value	Description
accountid	no	string	n/a	This parameter can be omitted if the user is bound to an account.

Body Parameters

Key	Required	Value Type	Default Value	Description
user	yes	RestUserInput	null	Represents a RestUserInput object either in JSON or XML.

2.11.17.2 Fields

Available Fields

Key	Value Type	Description
id	string	The id of the user
name	string	The name of the user
email	string	The email of the user

 © 2017 Kofax

Key	Value Type	Description
phone	string	The phone of the user
state	UserState	The state of the user
settings	string	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	UserRoleName	A list of all roles the user possesses: Possible values: USER, TEAMMGR, ADMIN, SUPERUSER
password	string	The password of the user
timeZone	TimeZone	The time zone of the user

2.11.17.3 Response

The response status is 200 if the user was successfully updated, otherwise a Kofax SignDoc Status Code is returned together with the explaining message(s).

2.11.18 Delete a User

Deletes the existing user specified by account ID and user ID.

NOTE

For this request a valid authentication and Account Administrator rights are necessary.

URL:	**/rest/v5/users/{userid}**
Producible Formats:	JSON, XML
Return Formats:	JSON, XML
Request Header(s):	Accept: application/json, application/xml
HTTP Method:	DELETE

Example Request URL

```
DELETE https://localhost:8080/cirrus/rest/v5/users/user1
```

Example Headers

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

2.11.18.1 Parameters

Path Parameters

Key	Required	Value Type	Default Value	Description
userid	yes	integer	n/a	ID of the user who should be deleted

Query Parameters

Key	Required	Value Type	Default Value	Description
accountid	no	string	n/a	The account ID. This parameter can be omitted if the user is not bound to an account. This is the case for users with the role SUPERUSER.

2.11.18.2 Response

The response status is 200 if the user was successfully deleted, otherwise a Kofax SignDoc Code is returned together with the explaining message(s).

2.11.19 Invite or Reinvite a User

This request sends an or a reinvitation to a user specified by ID.

NOTE

For this request a valid authentication and Account Administrator or Super User rights are necessary.

URL:	**/rest/v5/users/{userid}/invitation**
Producible Formats:	JSON, XML
Return formats:	JSON, XML
Request Header(s):	Accept: application/json, application/xml

 © 2017 Kofax

HTTP Method:	POST

Example Request URL

```
POST https://localhost:8080/cirrus/rest/v5/users/user1/invitation
```

Example Headers

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

2.11.19.1 Parameters

Path Parameters

Key	Required	Value Type	Default Value	Description
userid	yes	integer	n/a	ID of user

Query Parameters

Key	Required	Value Type	Default Value	Description
accountid	no	string	n/a	The account ID. It's only necessary if a user with role SUPERUSER wants to invite or reinvite an account administrator.

2.11.19.2 Response

The response status is 200 if the user was successfully invited or reinvited, otherwise a Kofax SignDoc Code is returned together with the explaining message(s).

2.11.20 Reset Password

This request resets the password of a Server Administrator identified by the ID or email address (@ is encoded as %40). The account ID is necessary when resetting the password of another user (only applicable for Server Administrator).

NOTE

For this request a valid authentication and Super User rights are necessary.

URL:	**/rest/v5/users/{userid}/pwreset**
Producible Formats:	JSON, XML
Return Formats:	JSON, XML
Request Header(s):	Accept: application/json, application/xml
HTTP Method:	POST

Example Request URL

```
POST https://localhost:8080/cirrus/rest/v5/users/user1/pwreset
```

Example Headers

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

2.11.20.1 Parameters

Path Parameters

Key	Required	Value Type	Default Value	Description
userid	yes	string	n/a	ID of the Server Administrator

Query Parameters

Key	Required	Value Type	Default Value	Description
accountid	no	string	n/a	The account ID. This parameter can be omitted if the user is not bound to an account. This is the case for users with the role SUPERUSER.

2.11.20.2 Response

The response status is 200 if the password was successfully reseted, otherwise a Kofax SignDoc Status Code is returned together with the explaining message(s).

 © 2017 Kofax

2.11.21 Add Role to User

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

NOTE

For this request a valid authentication and Account Administrator or Super User rights are necessary.

URL:	**/rest/v5/users/{userid}/role**
Producible Formats:	JSON, XML
Return Formats:	JSON, XML
Request Header(s):	Accept: application/json, application/xml
HTTP Method:	POST

Example Request URL

```
POST https://localhost:8080/cirrus/rest/v5/users/user1/role
```

Example Headers

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

2.11.21.1 Parameters

Path Parameters

Key	Required	Value Type	Default Value	Description
userid	yes	integer	n/a	ID of the user

Query Parameters

Key	Required	Value Type	Default Value	Description
accountid	no	string	n/a	The account ID. This parameter can be omitted if the user is not bound to an account. This is the case for users with the role SUPERUSER.
role	no	string	n/a	Name of the new role: ADMIN, TEAMMGR

2.11.21.2 Response

The response status is 200 if the role was successfully added, otherwise a Kofax SignDoc Status Code is returned together with the explaining message(s).

2.11.22 Remove Role from User

This request removes a role from a user specified by ID. Valid roles are ADMIN or TEAMMGR.

NOTE

For this request a valid authentication and Account Administrator or Super User rights are necessary.

URL:	**/rest/v5/users/{userid}/role**
Producible Formats:	JSON, XML
Return Formats:	JSON, XML
Request Header(s):	Accept: application/json, application/xml
HTTP Method:	DELETE

Example Request URL

```
DELETE https://localhost:8080/cirrus/rest/v5/users/user1/role
```

Example Headers

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

2.11.22.1 Parameters

Path Parameters

Key	Required	Value Type	Default Value	Description
userid	yes	integer	n/a	ID of the user

 © 2017 Kofax

Query Parameters

Key	Required	Value Type	Default Value	Description
accountid	no	string	n/a	The account ID. This parameter can be omitted if the user is not bound to an account. This is the case for users with the role SUPERUSER.
role	no	string	n/a	Name of the role: ADMIN, TEAMMGR

2.11.22.2 Response

The response status is 200 if the role was successfully removed, otherwise a Kofax SignDoc Status Code is returned together with the explaining message(s).

3 Kofax SignDoc Plugins

Plugin Development
Plugin Packaging
Deployment
Configuration
Available Plugins

3.1 Plugin Development

Plugin Interface
Implementation
How SignDoc Uses Plugins

3.1.1 Plugin Interface

SignDoc plugins support the ‘event’ plugin interface also implemented in SignDoc Web. In addition to that, they support the definition of plugin configuration data and the parameter description.

A plugin must implement the following interfaces:

IPlugin

The *de.softpro.sppluginif.IPlugin* interface defines the general plugin parameters. This includes the:

- Plugin id – used to identify the plugin. This has to be unique.
- General plugin information – like vendor, description, copyright.
- Error message information – provide localized information for a particular error.
- Injection point for plugin configuration information – is called by the application with the plugin configuration data when the plugin is instantiated.

IEventPlugin

The *de.softpro.sppluginif.IEventPlugin* interface defines the calling procedure for event based plugins. A plugin defines which events it supports by returning a list with *supportedEvents*. The application will then post an event to *eventCallback* providing a parameter map with event specific content and getting a result map, also with event specific content.

Each event definition will also provide a list of supported input and output parameters.

IConfigurablePlugin

The *de.softpro.cirrus.plugins.IconfigurablePlugin* interface defines how a SignDoc plugin can make its configurable parameters known to the application.

The application will use *getSettingDescriptions* to query what configuration settings the plugin supports. The plugin will return an array of *PluginSettingDescription*, where each element describes one setting:

- The name used to identify the setting.
- The description of the setting.
- If the setting is mandatory or optional.
- A Java regular expression that can be used to validate the setting value.

This information can be used by the application to provide an administrator GUI to allow plugin configuration. It is also used by the configuration service to validate entries.

The setting descriptions have to be returned localized, in the locale requested.

3.1.2 Implementation

It is recommended to use the abstract classes provided where applicable. The *AbstractEventPlugin* can be used as a basis for event plugins, if settings are needed, the *IConfigurablePlugin* interface has to be implemented.

For exceptions thrown by the plugin, use *de.softpro.sppluginif.PluginException* and error numbers defined in *de.softpro.sppluginif.EPluginMsgs* as far as possible. Even though you can define your own exception class and error number range, the application will only be able to react in an individual manner to exceptions it knows about. All custom exceptions will be treated the same as a *PLUGIN_UNKNOWN_EXCEPTION*.

3.1.3 How SignDoc Uses Plugins

- For a plugin to be used, the system administrator (role SUPER) has to add a plugin to the loadlist. Once the loadlist is changed with the configuration service, the application will dynamically reload the plugins in the loadlist.

 © 2017 Kofax

- For each plugin on the loadlist, the application will use the *IConfigurablePlugin* interface, if it is implemented, *PluginSettingDescription* returned, the configuration service will generate a configuration description according to the values returned.
  It will also generate an enabled setting for the plugin id.
- The system administrator and/or the account administrators can then enable the plugin globally or for specific accounts by setting the *plugin.enabled.<pluginId>* configuration setting to true. For every enablement an instance of the plugin will be created. If the plugin is enabled globally, only one instance will be created that is accessible for all accounts. If the plugin is enabled on an account specific basis, each account will use its own instance. This is also true if an account overrides the global setting.
- They can also provide the configuration information according to the settings provided by the plugin. This information can be account specific.
- The application will inject the configuration information via *injectPluginConfiguration* from the *IPlugin* interface. This is done for every instance of the plugin on an account specific basis.
- The plugin can now be used. The application will post supported events to the plugin using *eventCallback* from the *IEventPlugin* interface.

3.2 Plugin Packaging

Depending on the external dependencies plugins can be packaged either as a standalone class file, a JAR file or a ZIP file.

Class File

Delivering the plugin as a class file is feasible if the plugin does not have any dependencies. The class file needs to be deployed in the CLASSPATH or configured plugin load path.

JAR File

Packaging the plugin in a JAR file is feasible if the plugin only has local dependencies that can be packaged in the JAR file, or if the external dependencies are available to SignDoc via the CLASSPATH.

ZIP File

Packaging the plugin in a ZIP file lets you include additional dependencies to be delivered together with the plugin. The structure of the ZIP file is shown below.

```
MyPlugin.zip
|
+-- /classes
|       (simple class, properties, xml files)
|
+-- /lib
        (jar files)
```

The name of the ZIP file should be unique and reflect your plugin scope, since all ZIP files will be extracted to directories matching the name of the ZIP file. Ideally the ZIP file name will be the plugin id (without any special characters).

The plugin class path overrides the web-app classpath. This means that if a class of the plugin package references a class, and the same class exists in the server context and the plugin package. The class of the plugin package will be used.

3.3 Deployment

Plugins can be deployed in the application classpath (usually for internal plugins) or in the plugin directory.

The plugin directory is set via the plugin.directory configuration value (via the configuration service) and defaults to CIRRUS_HOME/plugins. Depending on the type of packaging chosen, the deliverable will have to be deployed in different places inside the plugins directory structure.

ZIP files will be deployed inside the plugins directory. Once the application starts, it will unpack the ZIP files and create a subdirectory with the name of the ZIP file.

For JAR or CLASS file packaging, the application will create a default directory, where all individually packaged plugins can be deployed. In all cases the structure below those directories is the same and will contain two subdirectories:

classes .class files should be deployed here.

lib .jar files should be deployed here.

3.4 Configuration

All plugin related configuration is done via the configuration service. The service is accessible via the REST API.

To be available for configuration and use a plugin must be loaded and enabled. Enabling a plugin can be done on an account basis.

3.4.1 Loading the Plugin

Initially a plugin has to be loaded. Upon load, it becomes available and the plugin configuration information is added to the configuration service. To load a plugin it has to be added to the loadlist. This operation is only accessible to a system administrator (role SUPER).

To add the sample notification plugin to the loadlist, the loadlist would be set to:

```
plugin.loadlist =
   de.softpro.cirrus.plugins.notification.NotificationSMSClickatell
```

If more than one class is added to the loadlist, the class names will be separated by ‘,’.

A sample JSON structure used by the REST API to set the loadlist would look like:

```
{
 "list": [
   {
     "k": "plugin.loadlist",
     "v": "de.softpro.cirrus.plugins.notification.NotificationSMSClickatell"
   }
 ]
}
```

Upon adding a plugin to the loadlist, the configuration service will load the plugin and add the plugin configuration elements to the list of available configuration items. It will also generate an enabled setting letting you enable or disable a plugin.

 © 2017 Kofax

3.4.2 Enabling a Plugin

Once a plugin is loaded, it can be enabled or disabled on an account basis. Loading a plugin will generate a configuration setting *plugin.enabled.<PluginId>* where *<PluginId>* is the plugin id you assigned.

As any configuration setting, this setting can be set on a global or on an account basis. The default value is false. The global value can be changed by a system administrator (role SUPER), while the account specific setting can be changed by an account administrator (role ADMIN).

When evaluating the setting the following order is used:

1. An account specific setting takes precedence over any other setting.
2. A global setting is used if no account specific setting is available for that particular account.
3. The default value is used if no setting is configured.

Thus, the system administrator can enable a plugin for all accounts by setting the enabled setting to true without using an account id. If the plugin is to be enabled only for specific accounts, the enabled setting would be set to true for those accounts, by providing the account id during the process.

A JSON structure for the enabled setting of the sample notification plugin would look like:

```
{
 "list": [
   {
     "k": "plugin.enabled.NotificationSMSClickatell",
     "v": "true"
   }
 ]
}
```

3.4.3 Configuring a Plugin

Once a plugin is loaded and enabled, it can be configured. As with the enabled setting above, the plugin configuration is also account specific.

Settings can be made on a global basis, for all accounts, or on an account specific basis. Account specific settings take precedence over global settings.

The configuration information available will be provided by the plugin upon instantiation.

A JSON structure for the notification sample plugin will look like:

```
{
 "list": [
   {
     "k": "plugin.cfg.NotificationSMSClickatell.url",
     "v": "http://smscatcher.sdlabs.de:8080/sendmsg"
   },
   {
     "k": "plugin.cfg.NotificationSMSClickatell.userid",
     "v": "test_user"
   },
   {
     "k": "plugin.cfg.NotificationSMSClickatell.password",
     "v": "test_password"
   },
   {
     "k": "plugin.cfg.NotificationSMSClickatell.apiid",
     "v": "1234567"
   }
 ]
}
```

If plugin configuration changes, the plugins will be reloaded and the configuration information injected. This applies to all servers sharing the configuration service.

3.5 Available Plugins

This paragraph describes the available plugin interfaces and their use in SignDoc.

The plugins will all implement the interfaces described above. They will support one or more events (see AbstractPluginEvent) that will trigger plugin actions or query plugin data.

3.5.1 Notification Plugin

The notification plugin interface is used to access a notification service to send out notification data. Currently this is the two factor authentication code a signer has to enter when an authentication code is required.

The sample implementation uses a SMS service (Clickatell) to send out the notification information via SMS.

Supported Events

The notification service supports two events:

- A notification parameter event used to query the parameter information available.
- A notification event used to actually send a notification

The application will usually first issue a parameter event to check what parameters a particular notification channel needs (in case of a SMS notification for instance the phone number). It will generate input fields for the parameters described by the plugin to capture the necessary parameters.

Once a signer needs to be notified, the application will issue a notification event, providing the parameters captured above.

Notification Parameter Event

de.softpro.cirrus.plugins.event.NotificationParametersEvent describes the notification parameters event. This event is used to query the parameters of a particular notification service.

The event input parameters describe what is requested:

IN_INFORMATION_LIST	The list of information to be queried. Can be: • IN_INFO_TYPE_DESCRIPTION (return the description of the notification service ()) • IN_INFO_MAX_MESSAGE_SIZE (return the maximum message size that can be sent out in one message based on the current plugin configuration) • IN_INFO_PARAMETERS (return the parameter descriptions needed to use the notification event) If no selection list is supplied, all of the above will be returned.

 © 2017 Kofax

IN_LOCALE	The locale the information should be returned in (IETF BCP 47 tag). If not specified the default locale is English.

The event will return the requested information as a list of output parameters containing OUT_NOTIFICATION_TYPE_DESCRIPTION, OUT_MAX_MESSAGE_SIZE and OUT_PARAMETERS. In case of OUT_PARAMETERS the parameter will contain a list of NotificationTargetParameterDescription objects, that each describe one input parameter to the notification event:

- name
- help text
- placeholder
- validation regular expression (used to validate the input)

The information above should be returned in the language specified by the IN_LOCALE parameter.

Notification Event

de.softpro.cirrus.plugins.event.NotificationEvent describes the event issued to actually trigger a notification.

The event input parameters are:

IN_TARGET	A map of target parameters described by the list retrieved by the previous event.
IN_MESSAGE	The message to be sent.

There are no output parameters.

3.5.2 Trusted Service Provider

The trusted service provider interface is used to add a TSP digital signature to the document if a signer is registered with the TSP.

This interface is used both by SignDoc Standard and SignDoc Web, which accounts for some peculiarities with the configuration. In SignDoc Standard the *IConfigurablePlugin* interface and the account specific instantiation is used to inject account specific configuration data via *IPlugin*. SignDoc Web does not support account specific configuration. Therefore, the event interfaces will use an optional settings parameter that SignDoc Web will pass with every call and that will override general settings. Thus, if a settings parameter is present, it should be given precedence over any settings injected via *IPlugin* at plugin instantiation.

Supported Events

The trusted service provider interface supports three events:

- *TSPInfoEvent* provides information about the TSP provider the plugin implements.
- *TSPValidationEvent* used to validate the credentials of a signer for a specific type of digital signature, before the actual signing takes place.

- *TSPSigningEvent* used to actually sign a document via the trusted service provider.

In case a specific provider does not support the validation step, the validation event does not need to be supported.

TSP Info Event

de.softpro.cirrus.plugins.event.tsp.TSPInfoEvent returns information specific to the TSP provider this plugin supports:

`IN_LOCALE`	Optional. The locale the information should be returned in (IETF BCP 47 tag). If not specified the default locale is English.
`IN_SETTINGS`	Optional. If present, the settings map will completely override the settings provided at plugin instantiation via *IPlugin.injectPluginConfiguration*. This is necessary, since SDWeb has no account specific plugin mechanism, thus the account specific parameters need to be passed for each call. The plugin has to be able to re-initialize on a 'by call' basis.

The output will provide provider specific information needed to display input and information pages related to the TSP validation and signing process:

`OUT_PROVIDER_NAME`	The (localized) name of the trusted service provider implemented by this plugin.
`OUT_PROVIDER_INFO`	Provider specific information that will be used when displaying input pages (descriptions, help, registration URL). This returns a map of settings described below.
`OUT_CREDENTIALS_DESCRIPTION`	The list of validation credentials needed by the TSP to perform a validation and/or signing. See below.

Currently supported provider info fields are:

- *PI_VALIDATION_TEXT* ("validation_text"): An optional text that describes the provider specific validation procedure.
- *PI_SIGNING_TEXT* ("signing_text"): An optional text that describes the provider specific signing procedure.
- PI_HELP_TEXT ("help_text"): An optional text that provides help and background information regarding the TSP provider.
- *PI_REGISTRATION_URL* ("registration_url"): An optional URL to the provider registration page where a signer can register a new user with this provider.

Each validation credential description element (*de.softpro.cirrus.plugins.event.tsp.TSPParameterDescription*) will consist of:

- A parameter id.
- A label to be shown for the entry field.
- A description (to be provided as a help text).
- A type.
- An indicator if the parameter is optional or mandatory.
- A placeholder text to be used in the entry field if needed.

 © 2017 Kofax

- A Java regular expression to be used to validate the user input.

The calling application can use the TSP info event to query the plugin on the information needed. The TSP name and the validation text will be displayed in the validation window, together with a list of entry fields defined by the validation credentials descriptions. If a registration URL is present, the application will display it, as part of a message where new users can create credentials if they are not yet registered.

TSP Validation Event

de.softpro.cirrus.plugins.event.tsp.TSPValidationEvent describes the actual validation call. The input parameters are:

`IN_CREDENTIALS`	The credentials, according to the information provided by the info event. Credentials are provided as a map with the parameter id as a key and the value given for that parameter. To avoid exceptions, parameters should be validated with the regular expression returned by the info event.
`IN_SIGNATURE_TYPE`	The type of the digital signature requested. Currently *BASIC*, *ADVANCED* or *QUALIFIED*.
`IN_SETTINGS`	Optional. If present, the settings map will completely override the settings provided at plugin instantiation via *IPlugin.injectPluginConfiguration*. This is necessary, since SDWeb has no account specific plugin mechanism, thus the account specific parameters need to be passed for each call. The plugin has to be able to re-initialize on a 'by call' basis.

The output only provides a true / false condition depending on the validation outcome:

`OUT_RESULT`	A boolean value indicating the result of the validation. True denotes a successful validation of the credentials for the specified signature type.

In case of processing errors, an appropriate exception will be thrown.

TSP Post Document Signature Event

de.softpro.cirrus.plugins.event.tsp.PostDocumentSignatureEvent starts the signing process with the TSP provider. The input parameters are:

`IN_CREDENTIALS`	The credentials, according to the information provided by the info event. Credentials are provided as a map with the parameter id as a key and the value given for that parameter. To avoid exceptions, parameters should be validated with the regular expression returned by the info event.
`IN_AUTHENTICATION_TOKEN`	An authentication token, if available. In case the TSP uses a short lived authentication token to authenticate the service, the token can be passed here.
`IN_SIGNATURE_TYPE`	The requested signature type (BASIC, ADVANCED or QUALIFIED).

IN_DOCUMENT	The document to be signed (byte array).
IN_DOCUMENT_NAME	The name of the document to be signed.
IN_REDIRECT_URL_OK	URL to be redirected to if signing was successful.
IN_REDIRECT_URL_CANCEL	URL to be redirected to if signing has been canceled.
IN_REDIRECT_URL_ERROR	URL to be redirected to if a signing error occurred.
IN_SETTINGS	Optional. If present, the settings map will completely override the settings provided at plugin instantiation via IPlugin.injectPluginConfiguration. This is necessary, since SDWeb has no account specific plugin mechanism, thus the account specific parameters need to be passed for each call. The plugin has to be able to re-initialize on a 'by call' basis.

The output data includes:

OUT_AUTHENTICATION_TOKEN	A new authentication token after successful authentication, in case the TSP uses an authentication token mechanism.
OUT_SIGNATURE_PROCESS_TOKEN	The signature token returned by the TSP after initiating the document signing. This token is used to identify this particular signing process.
OUT_DOCUMENT_VERIFICATION_URL	The TSP document verification URL. The signing client will redirect to this TSP URL letting the signer authenticate with the TSP service and sign the document.

TSP Get Document Signature Event

de.softpro.cirrus.plugins.event.tsp.GetDocumentSignatureEvent is used to retrieve a signed document from the TSP previously sent for signing.

The input parameters are:

IN_SIGNATURE_PROCESS_TOKEN	The signature token that identifies this signing process (received by PostDocumentSignatureEvent).
IN_AUTHENTICATION_TOKEN	An authentication token, if available. In case the TSP uses a short lived authentication token to authenticate the service, the token can be passed here.
IN_SETTINGS	Optional. If present, the settings map will completely override the settings provided at plugin instantiation via IPlugin.injectPluginConfiguration. This is necessary, since SDWeb has no account specific plugin mechanism, thus the account specific parameters need to be passed for each call. The plugin has to be able to re-initialize on a 'by call' basis.

 © 2017 Kofax

The output data includes:

OUT_AUTHENTICATION_TOKEN	A new authentication token after successful authentication, in case the TSP uses an authentication token mechanism.
OUT_DOCUMENT	The content of the signed document (byte array).
OUT_DOCUMENT_NAME	The name of the document being returned.

4 Appendix

Contact Information
Trademarks
Copyright Notice

4.1 Contact Information

The Kofax technical support team will be happy to assist you.

If you need support with regards to purchased Signature products please contact us via Kofax Customer Portal:

https://techsupport.kofax.com

4.2 Trademarks

- SignDoc, FraudOne, SignPlus, SignWare, SignInfo, SignCheck, SIVAL, SignPad, SignSecure are registered trade marks of Kofax Inc. or its affiliates – all rights reserved.
- All other brand and product names can be trademarks, service marks or other intellectual property of the respective owners who reserve all related rights.

4.3 Copyright Notice

© 2017 Kofax. All rights reserved.

Kofax is a trademark of Kofax Inc., registered in the U.S. and/or other countries. All other trademarks are the property of their respective owners. No part of this publication may be reproduced, stored, or transmitted in any form without the prior written permission of Kofax.

Index

A

account
add team 122
add user 144
create 15
delete 19
delete team 125
import license 22
update 18
Account Administrator
role 13
account requests 15
accountFilter 21
accountid 23, 131, 132, 145, 148, 149, 151, 153, 154, 155, 156, 157
add
checkbox to document 44
document to package 31
reminder to package 105
role to user 156
signature field to document 52
signer to package 110
team to account 122
text field to document 59
user to account 144
user to team 128
all accounts
get 23
all fields
get 50
all server administrators
get 137
all users
get 147
audit trail
get 97
auditTrailOptions 68
authentication token 10
create 129
refresh 133
authentication via API key 12
autoprep 32

C

checkbox 45
add to document 44
update with form data 47
checkboxFields 36
checkboxid 47, 48
complete structure
signing package 86
configuration
plugin 161
configuration requests 25
configuration setting
delete 25
get 26
set 28
configuration setting description
get 29
configuring
plugin 162
create
account 15
authentication token 129
new signing package 66
password recovery for user 132
server administrator 135
signing authentication token 119
signing package from scratch 82
signing package from template 83
signing package from template with meta data 83
signing package with Word 85
signing packages with Word 80
create signing package
options 69
credentials 132
current server administrator
get 134
current user
get 146

D

delete
account 19
configuration setting 25
document 37
field 51
reminder 109
server administrator 138

 © 2017 Kofax

delete
signer 115
signing package 88
team from account 125
user 152
deployment
plugin 161
document
add checkbox 44
add signature field 52
add text field 59
add to package 31
delete 37
update 34
document requests 31
documentid 35, 37, 38, 45, 47, 48, 51, 52, 53, 56
documentMessage 36

E

enabling
plugin 162
ERestAccountState 17
ERestDocumentFieldType 51
ERestDocumentFormat 33
error code information 10
event
trigger 41
event requests 40
expirationDate 68

F

field
delete 51
field requests 43
fieldFilter 51
fieldid 52
fieldName 48, 57, 62
fileName 36
form data
update a checkbox 47
update existing signature field 91
update signature field 57
update text field 62

G

get
all accounts 23
all fields 50
all server administrators 137
all users 147
audit trail 97
configuration setting 26
configuration setting description 29
current server administrator 134
current user 146
list of packages 100
number of accounts 24
number of users 131
single account 20
single checkbox 46
single document 38
single server administrator 139
single signature field 55
single signer 113
single singning package 93
single text field 64
single user 149
time zones 120
version 121

I

implementation
plugin 159
import
license to account 22
interface
plugin 158
invite
server administrator 142
user 153

L

license 23
import to account 22
list of packages
get 100
loading
plugin 161

M

mailMessage 68
mailSubject 68
maxLength 61, 63
metadata

metadata
create signing package from template 83
multipleLine 61, 62, 63

N

newAccount 19
notification
plugin 163
number of accounts
get 24
number of users
get 131

O

options
create signing package 69

P

package
add document 31
add reminder 105
add signer 110
packageid 32, 35, 37, 38, 45, 47, 48, 51, 52, 53, 56
packaging
plugin 160
password
reset 154
reset for server administrator 143
password recovery
create for user 132
plugin
available 163
configuration 161
configuring 162
deployment 161
enabling 162
implementation 159
loading 161
notification 163
packaging 160
plugins
use 159
prepare
signing session 102
processingType 68
properties overview
signing package 71
pugin
interface 158

R

readOnly 45, 48, 49, 58, 62
refresh
authentication token 133
reinvite
server administrator 142
user 153
reminder
add to package 105
delete 109
update 107
reminder requests 105
remove
role from user 157
user from team 126
representation formats 12
reset
password 154
password of server administrator 143
REST error response 9
REST interface 9
REST URL 9
RestAccountInput 16, 19
RestDocument 33
RestMsg 9
RestMsgList 9
RestReminder 33
RestSignatureFieldInput 53
RestSignerInput 71
RestSignerOutput 114
restSigningPackageInput 67
RestTextFieldInput 60
RestUserInput 136
role 156, 157
Account Administrator 13
add to user 156
remove from user 157
Super User 13
Team Manager 13
User 13

S

schedule
signing package 101

 © 2017 Kofax

server administrator
create 135
delete 138
invite 142
reinvite 142
reset password 143
update 140
set
configuration setting 28
sigle signing package
get 93
signature field
add to document 52
update with form data 57, 91
signaturefieldid 56
signatureFields 36
signer
add to package 110
delete 115
update 116
signer requests 110
signerId 45, 49, 58
signing authentication token
create 119
signing package
complete structure 86
create from scratch 82
create from template 83
create from template with metadata 83
create new 66
create with Word 85
delete 88
properties overview 71
schedule 101
update 89
signing package requests 66
signing packages
create with Word 80
signing session
prepare 102
signingModeOption 68
signingModeOptions 54, 58
single account
get 20
single checkbox
get 46
single document
get 38
single server administrator
get 139
single signature field
get 55
single signer
get 113
single text field
get 64
single user
get 149
startDate 68
swagger UI 12
system requests 120

T

team
add to account 122
add user 128
delete from account 125
remove user 126
update 124
team requests 122
teamid 128
template
create signing package 83
create signing package with metadata 83
text field
add to document 59
update with form data 62
textFields 36
time zones
get 120
TimeZone 136, 141, 145, 151
trigger
event 41
trusted service provider 164

U

update
account 18
checkbox with form data 47
document 34
existing signature field with form data 91
reminder 107
server administrator 140
signature field with form data 57
signer 116
signing package 89
team 124
text field with form data 62

update
user 150
updDocument 35
use
plugin 159
user
add role 156
add to account 144
add to team 128
create password recovery 132
delete 152
invite 153
reinvite 153
remove from team 126
remove role 157
update 150
user requests 129
user roles 13
userid 138, 140, 141, 143, 144, 149, 151, 153, 154, 155, 156, 157
UserRoleName 136, 141, 145, 151
users
get number 131
UserState 136, 141, 145, 151

V

version
get 121

W

Word
create signing package 85
create signing packages 80

 © 2017 Kofax