# Table of Contents

**Preface** ................................................................................................................................. 5  
  Related documentation ........................................................................................................ 5  
  Training .................................................................................................................................. 5  
  Getting help with Kofax products ....................................................................................... 5  
  Definitions ............................................................................................................................. 6  

**Chapter 1: Licensing** .............................................................................................................. 7  
  License handling .................................................................................................................. 7  
  SignDoc license .................................................................................................................... 7  
  Account license .................................................................................................................... 8  
  Global license ....................................................................................................................... 8  
  Reset license (special license) .............................................................................................. 9  

**Chapter 2: Logging** ................................................................................................................ 10  
  On-premise logging ............................................................................................................. 10  
  Request logging ................................................................................................................... 11  

**Chapter 3: Configuration** ..................................................................................................... 13  
  Configuration service ......................................................................................................... 13  
    Configuration levels ........................................................................................................ 14  
    Change configuration options ....................................................................................... 14  
  Configuration files ............................................................................................................ 19  
    Configure the ‘autoprepare’ functionality ................................................................ 19  
    Fonts used in the final document ................................................................................. 20  
  Configuration values ......................................................................................................... 20  
    System ............................................................................................................................ 20  
    Documents and packages ............................................................................................. 22  
    Security .......................................................................................................................... 23  
    Mail ................................................................................................................................. 24  
    Plugins ........................................................................................................................... 29  
    Client .............................................................................................................................. 29  
    Advanced signing settings ......................................................................................... 31  

**Chapter 4: Plugins** ............................................................................................................... 33  
  Enable plugins .................................................................................................................... 33  
  Plugin load list .................................................................................................................... 33  
  Plugin enabled ..................................................................................................................... 33  
  Configure plugins .............................................................................................................. 34
Sample workflow to enable a plugin.................................................................................................................................34
Enable a plugin globally......................................................................................................................................................34
Enable a plugin for a specific account...............................................................................................................................37
Core plugins..........................................................................................................................................................39
  ClamAV virus scan plugin.............................................................................................................................................39
  KTA state change plugin.................................................................................................................................................41
  TSP plugin.................................................................................................................................................................42
  SMS notification plugin..................................................................................................................................................45

Chapter 5: Monitoring............................................................................................................................................................50
Overview...........................................................................................................................................................................50
  Monitoring the KSD cluster............................................................................................................................................51
  Monitoring protocol.....................................................................................................................................................51
  Metrics being collected..................................................................................................................................................52
  Monitoring server.........................................................................................................................................................52
SignDoc Standard application metrics..................................................................................................................................52
  Metric overview...........................................................................................................................................................53
  Configuration...............................................................................................................................................................54
  Metric description........................................................................................................................................................54
Example setups......................................................................................................................................................................58
  Monitoring service sample setup......................................................................................................................................58
  Collectd sample setup for collecting system metrics....................................................................................................59
Glossary.................................................................................................................................................................................61

Chapter 6: Tenant-specific URL..............................................................................................................................................63
Preface

This guide provides information on SignDoc Standard licensing, logging, configuration, plugins and monitoring.

Related documentation

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

https://docshield.kofax.com/Portal/Products/en_US/SD/2.2.0-aq3yp7tk0j/SD.htm

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

• Help for Kofax SignDoc Standard
• Help for Kofax SignDoc Standard Administration Center
• Help for Signing Documents with Kofax SignDoc
• Kofax SignDoc Standard Developer’s Guide
• Kofax SignDoc Standard Installation Guide

Training

Kofax offers both classroom and computer-based training that will help you make the most of your Kofax SignDoc solution. Visit the Kofax website at www.kofax.com for complete details about the available training options and schedules.

Getting help with Kofax products

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

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

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

The Kofax Knowledge Base provides:

• Powerful search capabilities to help you quickly locate the information you need.
  Type your search terms or phrase into the Search box, and then click the search icon.
• Product information, configuration details and documentation, including release news.
  Scroll through the Kofax Knowledge Base home page to locate a product family. Then click a product
  family name to view a list of related articles. Please note that some product families require a valid
  Kofax Portal login to view related articles.
• Access to the Kofax Customer Portal (for eligible customers).
  Click the Customer Support link at the top of the page, and then click Log in to the Customer Portal.
• Access to the Kofax Partner Portal (for eligible partners).
  Click the Partner Support link at the top of the page, and then click Log in to the Partner Portal.
• Access to Kofax support commitments, lifecycle policies, electronic fulfillment details, and self-service
  tools.
  Scroll to the General Support section, click Support Details, and then select the appropriate tab.

Definitions

INSTALLDIR
The directory with the unpacked signdoc-standard-2.1.0.zip file. See Related documentation, SignDoc
Standard Installation Guide, chapter "Quickstart procedure".

CIRRUS_HOME, SDWEB_HOME
The directories of the web applications that compose SignDoc Standard. Starting with SignDoc
Standard 2.1.0, these home directories are consolidated by default in one single directory. See Related
documentation, SignDoc Standard Installation Guide, chapter "Directories".
Chapter 1

Licensing

License handling

To use SignDoc Standard it is required to install a valid SignDoc license for SignDoc Standard.

For on-premise installations the SignDoc license file (KofaxSignDoc.key) will be provided by the Kofax Order Fulfillment team based on a sales order. You will get a set of license files which is specially made out for the customer.

SignDoc license

To run SignDoc an appropriate license key file has to be installed. The license key contains information about the permissions. The key file can be opened in a text editor. The order of the license defines the permissions.

A SignDoc license can be installed only if the following requirements are met:

• It must be a valid SignDoc license
• The license is not expired

Starting with SignDoc 2.1.0.1 a SignDoc license (KofaxSignDoc.key) can either be used for an individual account or for all accounts of an installation.

Example for the content of a SignDoc license key:

```
h:SPLM2 4.10
i:ID:9923379
i:Product:KofaxSignDoc
i:Manufacturer:Kofax Deutschland GmbH
i:Customer:Dummy corporation
i:Version:99
i:OS:all
a:product:unlimited
a:signware:unlimited
a:sign:2018-12-31
a:capture:unlimited
a:render:unlimited
ps:ACCOUNT_INFORMATION:F4D22WO65F
ps:CustomerCompany:Dummy corporation
ps:COUNTER_RENEWAL_PERIOD:MONTHLY
pi:LICENSE_TYPE:1
pi:MAX_SIGNING_PACKAGES:100000
pi:MAX_USERS:100
```
Explanation of application-related license parameter:

- `a:sign:2018-12-31` Contains the date of expiry.
- `pi:MAX_SIGNING_PACKAGES:100000` Describes the number of licensed signing packages.
- `pi:MAX_USERS:100` Shows the maximum number of SignDoc users that can be created.
- `ps:ACCOUNT_INFORMATION:F4D22WO65F` Is a customer unique string and is important if you need an upgrade of the license, for example if you need more packages or if you want to increase the number of users. Only a license with the same `ps:ACCOUNT_INFORMATION` entry can be used for a subsequent license import for the account. Once installed, this `ps:ACCOUNT_INFORMATION` is dedicated for a specific account, that means, that a license with the same `ps:ACCOUNT_INFORMATION` cannot be used for more than one account.
- `ps:COUNTER_RENEWAL_PERIOD:MONTHLY` Causes a monthly package counter reset (default is YEARLY).

**Account license**

A SignDoc license will become an account license when the license file is applied for an individual account of the SignDoc installation. The account license includes permissions which are only valid for this specific SignDoc account. Any other accounts in the system need a separate license in each case. An account license must be installed during account creation. It is not possible to install an account license separately from the account creation. It is only possible to upgrade the license afterwards.

An account license can be installed for the first time only by a SignDoc server administrator during account creation. An upgrade of an account license can be installed by a SignDoc server administrator or by an account administrator.

An account license can be upgraded only with another license with the same `ps:ACCOUNT_INFORMATION` entry. Additionally, the following is required:

- It must be a valid SignDoc license
- The license is not expired
- The number of already processed packages must not exceed the number of licensed packages
- The number of already existing account specific users must not exceed the number of licensed users

**Global license**

A SignDoc license will become a global license when the license file is not applied for a specific account. In this case, the license can be used as system-wide license which is valid for all accounts. The included permissions are valid for all accounts without the necessity of an individual license for each account. The licensed number of packages and/or users are shared between all accounts.
A global license can be installed by a SignDoc server administrator before any account is created. If a global license is installed an arbitrary number of accounts can be installed. No additional account-specific licenses are needed.

**Note** If a global license is installed once it is not possible to return back to account-specific licenses! In general, if a global license is installed, any installed account-specific licenses are invalidated or rather removed.

A global license can be upgraded only with another license with the same `ps:ACCOUNT_INFORMATION` entry. Additionally, the following is required:

- It must be a valid SignDoc license
- The license is not expired
- The number of already processed packages must not exceed the number of licensed packages
- The number of already existing account specific users must not exceed the number of licensed users

**Use an installed account license as global license**

It is possible to use an applied account license as global license for the system. When the license key file will be applied again as global license the number of already processed packages for this account is used as base number of packages for the global license. All subsequent created signing packages (including templates) are count on base of this number. During installation of the global license it is checked whether the total number of all account-related users exceeds the number of licensed users. In this case the license cannot be installed.

**Apply a global license as replacement for already installed account licenses**

It is possible to install a new SignDoc license as global license even if you have installed any account-specific licenses.

**Reset license (special license)**

A reset license is a special license which must be requested from Kofax if the installed license is invalid or corrupted for any reason. Since this is an unrecoverable problem, it requires a special treatment. If an account license or a global license is corrupted it is required to ‘reset’ the license by this specific reset license. Only the import of a reset license allows the application to continue with an installation of a valid ‘normal’ license (account or global license).

It is not necessary (and also not possible) to use a reset license if the already installed license is valid or only expired or if any maximum (of users or packages) is reached. A reset license is a time restricted license file which cannot be used as production license.
Chapter 2

Logging

On-premise logging

SignDoc Standard is able to trace any application messages into a log. This log is independent from the audit trail which records package processing steps into the SignDoc Standard database.

SignDoc Standard uses an internal logging framework XjLog which uses Logback (http://logback.qos.ch/). Logback is intended as a successor to the popular log4j project.

By default SignDoc Standard creates log files on disk. Cirrus-specific log files are written to %CIRRUS_HOME%/logs.

%CIRRUS_HOME% is an environment variable which has to point to a valid directory. XjLog can be configured via XjLog.xml file which is available in %CIRRUS_HOME%/conf.

Cirrus writes its log by default into a file cirrus.log. The RollingFileAppender which is used for logging here uses FixedWindowRollingPolicy as rollingPolicy and SizeBasedTriggeringPolicy as triggeringPolicy. SizeBasedTriggeringPolicy looks at the size of the currently active file. If it grows larger than the specified size, it will signal the owning RollingFileAppender to trigger the rollover of the existing active file. With FixedWindowRollingPolicy cirrus.log is renamed to cirrus1.log if the maximum log file (currently 10MB) is reached. A new cirrus.log file is then created where new log entries are written. In the second rollover cirrus1.log is renamed as cirrus2.log and cirrus.log is renamed as cirrus1.log. The upper bound index is 10 (by default). If this index is reached this (oldest) log file cirrus10.log will be deleted and the previous (cirrus9.log) is renamed to cirrus10.log.

For better identifying the error logs are written to a separate subdirectory

%CIRRUS_HOME%/logs/error

The errors of each session are logged into a separate error log. The file name is either the current sessionid or the current timestamp if the error is independent from a specific session.

This kind of file logging is intended only for on-premise usage.
Request logging

**Important** SignDoc Standard

before version 2.1.0 was mainly configured with the configuration file cirrus.properties. This file moved to `INSTALLDIR\_conf_templates\cirrus.properties` with version 2.1.0.

Since SignDoc Standard 2.1.0, it is highly recommended to use the file `INSTALLDIR\service_configuration.properties` (instead of cirrus.properties) whenever it is required to configure SignDoc with a configuration file. Configurations set in this file are applied as Java System Property and have therefore highest precedence.

Each http request to SignDoc Standard and the appropriate response is logged.

For example, the browser call

```
http://beaker:8081/cirrus/
```
to the SignDoc Welcome page could be logged as follows in cirrus.log:

```
```

```
```

```
```

In the configuration file

```
%CIRRUS_HOME%/conf/cirrus.properties
```

the following settings can be set or changed for getting less or more detailed request and response information:

- Log all requests including parameters.
  
  cirrus.logging.all_requests=false

- Log all servlet requests including parameters.
  
  cirrus.logging.servlet_requests=true

- Log all REST requests including parameters.
  
  cirrus.logging.rest_requests=true
• Log all cookies.
cirrus.logging.request.all_cookies=false

• Log all JSESSIONID cookies.
cirrus.logging.request.jsession_cookies=false

• Log all request headers.
cirrus.logging.request_headers=true

• Log also body (if exists) of request (if the request is enabled for logging).
cirrus.logging.request_body=false

• Output limit of the requested body log entry (in bytes), set -1 for unlimited.
cirrus.logging.request_body_length=200

• Log all response headers.
cirrus.logging.response_headers=true

• Log response content (if available).
cirrus.logging.response_content=false

• Output limit of the response content log entry (in bytes), set -1 for unlimited.
cirrus.logging.response_content_length=200

• Include specific URIs (or parts from it) for request and response logging
  cirrus.logging.include_uri_list_request=
  Example
  cirrus.logging.include_uri_list_request=/LogService/,/ConfigurationService

• Exclude specific URIs (or parts from it) from request and response logging
  cirrus.logging.exclude_uri_list_request=
  Example
  cirrus.logging.exclude_uri_list_request=/LogService,/ConfigurationService

• Include specific URIs (or parts from it) for response content logging
  cirrus.logging.include_uri_list_response_content=
  Example
  cirrus.logging.include_uri_list_response_content=/services/,/rest/>

• Exclude specific URLs (or parts from it) from response content logging.
cirrus.logging.exclude_uri_list_response_content=

• Exclude specific requested resource files (file extension which must be at the end of the URI) from
request and response logging.
cirrus.loggingexclude_uri_list_file_suffix=.gif,.png,.jpg,.css,.js,.svg,.ico

**Prerequisite**

The log level of RequestLogFilter must be set to INFO in %CIRRUS_HOME%/conf/XjLog.xml (default):

```
<logger name="de.softpro.cirrus.web.helper.RequestLogFilter" level="debug"
        additivity="false">
    <appender-ref ref="CIRRUS" />
    <appender-ref ref="REDIS" />
</logger>
```
Chapter 3

Configuration

Important SignDoc Standard before version 2.1.0 was mainly configured with the configuration file cirrus.properties. This file moved to INSTALLDIR\_conf_templates\cirrus.properties with version 2.1.0.

Since SignDoc Standard 2.1.0, it is highly recommended to use the file INSTALLDIR\service_configuration.properties (instead of cirrus.properties) whenever it is required to configure SignDoc with a configuration file. Configurations set in this file are applied as Java System Property and have therefore highest precedence.

Starting with the release 1.3, some configuration options have been moved from configuration files to the configuration service.

Monitoring and the metrics processed can be configured in cirrus.properties configuration file with the following properties:

- **monitoring.host** The hostname or IP address of the monitoring server the metrics are sent to. If left empty, no monitoring data is sent (default).
- **monitoring.port** The port number where monitoring data is sent to. Defaults to 2003
- **monitoring.protocol** The protocol being used (TCP or UDP). Defaults to UDP.
- **monitoring.filter.include** Regular expression to specify which metrics form the available list should be reported. Defaults to all.
- **monitoring.filter.exclude** Regular expression specifying which metrics should be excluded from monitoring. By default, following statistic information is being excluded: count, m5_rate, m15_rate, max, min, mean_rate, p50, p75, p95, p98, p99, p999, stddev

Configuration service

The reasons for using a configuration service are:

- Changing configuration options on the fly, without a server restart.
- Setting configuration values individually on an account basis.
- Providing help and validation on configuration values via a GUI.
Configuration levels

One configuration value can be set on following levels:

- **Default value**
  The default value for a configuration option can be provided by the application. This value applies if no configuration is set by the user. This value cannot be changed.

- **Global (application-wide) value**
  This value can be set by the server administrator and applies to all accounts.

- **Account-specific value**
  This value can be set by an account administrator (permission dependent) or by the server administrator. It only applies to the account it has been set for. Not all configuration values can be set account specific.

The application uses following precedence when determining the value to apply for one configuration option and account:

1. If an account-specific value exists, it is used.
2. If no account-specific value exists, the global value is used if available.
3. If no global value exists, the default value is used.
4. If no default value is defined, no value is returned.

Change configuration options

All configuration service values are set using the REST API. Both the Administration Center and the Manage Client provide user interfaces to view and edit configuration values. They use the underlying REST API to apply the changes.

Configuration using the Administration Center

Server administrators use the Administration Center to change configuration values. They can change both global and account-specific values.

From the Administration Center starting page click the **System settings** link in the navigation panel to change global values.

The **System settings** menu is displayed which includes settings related to the system, documents and packages, plugins, security and signing.
The configuration options are grouped in categories shown on the left.

Each configuration option will have:

- A title
- The configuration option id
- A description of the configuration option

The entry field is used to edit the configuration option value. The configuration value on the current level (in this case global) is shown in normal text. If it is not set and a lower precedence level (in this case default) exists, that value will be shown grayed out.

A configuration value will be validated by the client before it is used.

Multiple configuration values can and should be set in one go. Related configuration values should be changed together. When everything is set, the **Save** button will store the new configuration on the server.

Account-specific configuration options can be set by first selecting the account:
Clicking **Edit** will take you to a similar view as above, where the account-specific settings can be edited.
Configuration using the Manage Client

Account administrators will use the Manage Client to change configuration settings. To get to the configuration settings, click Administration in the title bar:
The configuration options are grouped in categories shown on the left. The display and editing of individual configuration options is similar to the Administration Center.

**Configuration using the REST API**

All configuration options can also be set programmatically using the REST API.
Using the REST API is documented in the *SignDoc Standard Developer's Guide*, see Related documentation.

### Configuration files

**Important** SignDoc Standard before version 2.1.0 was mainly configured with the configuration file cirrus.properties. This file moved to INSTALLDIR\conf_templates\cirrus.properties with version 2.1.0. Since SignDoc Standard 2.1.0, it is highly recommended to use the file INSTALLDIR \service_configuration.properties (instead of cirrus.properties) whenever it is required to configure SignDoc with a configuration file. Configurations set in this file are applied as Java System Property and have therefore highest precedence.

#### Configuration options

- **cirrus.esign.consent**  This property defines the default e-sign consent text that is applied to every new account and can be changed later in the account administration section of the application. Default: 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.

- **cirrus.fail.login.max.attempts**  This property is used to determine after how many failed login attempts a user will be suspended. When a user is suspended due to reaching the maximum configured number of login attempts, an email will be send to that user. When a successful login is made, the user-specific number of failed logins is reset. Default: 10

- **cirrus.finaldocument.font.name**  With this property the font used in the final document is set. Default: DejaVu Sans

- **cirrus.tenant.url.supported**  In the cirrus.properties configuration file this setting determines whether tenant-specific URL handling is supported or not. Supported values: true, false. Default: false

For information on LDAP properties, see *SignDoc Standard Installation Guide*, chapter "Authentication LDAP*. See Related documentation.

### Configure the 'autoprepare' functionality

**Important** SignDoc Standard before version 2.1.0 was mainly configured with the configuration file cirrus.properties. This file moved to INSTALLDIR\conf_templates\cirrus.properties with version 2.1.0. Since SignDoc Standard 2.1.0, it is highly recommended to use the file INSTALLDIR \service_configuration.properties (instead of cirrus.properties) whenever it is required to configure SignDoc with a configuration file. Configurations set in this file are applied as Java System Property and have therefore highest precedence.
During preparation of a signing package the user may choose to 'autoprepare' a document after the upload. This means that for every defined signer a signature field is added to the document automatically.

Signature fields are placed in rows and columns starting at top/left margin. If a field would exceed the right margin it will be placed in a next row. If there is no space left on the page, an exception is thrown.

**Properties used for configuring the 'autoprepare' functionality**

Add configuration properties to \INSTALLDIR\service_configuration.properties configuration file.

- **cirrus.autoprepare.option** Autoprepare first or last page, defaults to first.
- **cirrus.autoprepare.margin.top** The top margin of the page where signature fields are placed.
- **cirrus.autoprepare.margin.left** The left margin of the page where signature fields are placed.
- **cirrus.autoprepare.margin.bottom** The bottom margin of the page where signature fields are placed.
- **cirrus.autoprepare.margin.right** The right margin of the page where signature fields are placed.
- **cirrus.autoprepare.padding.horiz** The space between rows of signature fields.
- **cirrus.autoprepare.padding.vert** The space between columns of signature fields.

All numeric values are in pixel units. The default value is always 10.

**Fonts used in the final document**

**Important** SignDoc Standard before version 2.1.0 was mainly configured with the configuration file cirrus.properties. This file moved to \INSTALLDIR\_conf_templates\cirrus.properties with version 2.1.0.

Since SignDoc Standard 2.1.0, it is highly recommended to use the file \INSTALLDIR\service_configuration.properties (instead of cirrus.properties) whenever it is required to configure SignDoc with a configuration file. Configurations set in this file are applied as Java System Property and have therefore highest precedence.

When building the final document, the application searches for compatible fonts in \INSTALLDIR\signdoc_home\fonts directory. Copy any fonts that might be used in the application in that location.

A preferred font can be specified by changing the value of the cirrus.finaldocument.font.name property. Add the configuration properties to \INSTALLDIR\service_configuration.properties.

**Configuration values**

**System**

This section lists the system settings.
General system settings

- Configuration cache check time
cirrus.config.cache_check_period
  The time interval in milliseconds to check if the configuration cache has to be updated.

- Certificate expiration warning threshold (days)
client.account.certificate.expiry.warn.threshold
  The threshold in days when the client will start warning of expiring certificate (1-365).

- License expiration warning threshold (days)
client.account.license.expiry.warn.threshold
  The threshold in days when the client will start warning on expiring account licenses (1-365).

REST API settings

- Timeout event request
cirrus.rest.event.request.timeout
  The maximum wait time in seconds (for long polling) if one or more events are requested via REST API
  (needed by the Manage Client).

- User authentication check time
cirrus.rest.event.user.authenticated.request.authid_check_period
  The time interval in seconds to check if the user was authenticated with another authentication id. The
  verification is needed by the Manage Client for auto logoff.

- Validate HTML input and reject, if invalid or dangerous
cirrus.rest.html_input.validate
  If HTML fragments contain invalid or dangerous constructs, the data will be rejected.

- Sanitize HTML input, if invalid or dangerous
cirrus.rest.html_output.sanitize
  If HTML fragments contain invalid or dangerous constructs, the data will be sanitized.

- General REST result set size limit
cirrus.rest.resultset.size.max.general
  General size limit of result sets for lists which are retrieved via REST API. The general settings can be
  overwritten by resource-specific settings.

- Package REST result set size limit
cirrus.rest.resultset.size.max.packages
  Size limit of result sets for package lists which are retrieved via REST API. If this configuration value is
  not set then the general setting cirrus.rest.resultset.size.max.general is used for result set limitation.

- Signer REST result set size limit
cirrus.rest.resultset.size.max.signers
  Size limit of result sets for signer lists which are retrieved via REST API. If this configuration value is not
  set then the general setting cirrus.rest.resultset.size.max.general is used for result set limitation.

- Signer REST result set size limit if a search criteria is being used
cirrus.rest.resultset.size.max.signers.autocomplete
  Size limit of result sets for signer lists which are retrieved via REST API in case one of the search
  criteria is being used. This is the case for the client autocomplete function.
• User REST result set size limit
  cirrus.rest.resultset.size.max.users
  Size limit of result sets for user lists which are retrieved via REST API. If this configuration value is not set then the general setting cirrus.rest.resultset.size.max.general is used for result set limitation.

• Require password to change a user’s email address
  cirrus.rest.user.email_change.require_password
  If enabled (i.e. set to true) the authenticated user must also supply the own password to change any user’s email address.

Documents and packages

This section lists the settings related to documents and packages.

• Source of the signer id which is assigned to a signature (line) field
  cirrus.document.prepare.msword.signatureline.signerid.source
  A Microsoft Word document can be uploaded to a signing package. The document can contain Word specific signature lines. A signature field is created in the converted pdf document for each signature line. In addition a signer is assigned to the new created signature field. The signer is either already available in the package or it will be created automatically. If the value is 'field_name' then the signer id is set from the signature line tag id (which is also used for the new created signature field). If the value of this setting is 'signer_name' then the source of the id for this signer is the 'Suggested signer' name from the Signature setup dialog (visible during insert signature line action). Note: The entered signer name must be conform to the allowed characters for an id [a-zA-Z0-9_—] and must not contain spaces. Otherwise a random UUID is set as signer id. Allowed values for this configuration settings are either field_name or signer_name.

• The maximum document size
  cirrus.document.prepare.size.max
  The maximum size of a document that can be uploaded.

• Maximum number of files for a supplemental document type
  cirrus.document.prepare.supplemental.file.max-number
  The maximum number of supplemental files that can be uploaded by a signer for one document type. Changing this number does not have an automatic impact on already configured document types or document type instances. This setting is only considered if you change or add a document type.

• Maximum number of supplemental document type instances
  cirrus.document.prepare.supplemental.type-instance.max-number
  The maximum number of supplemental document type instances that can be created for a signer.

• The script font for Click-to-Sign signatures
  cirrus.document.signing.c2s.font.script
  Click-to-Sign signatures need a "script font" that is used to render the signers's name. If the uploaded data is not usable it will be rejected.
  The default Click-to-Sign script font covers only a small subset (Western-Latin) of Unicode. If other characters are entered, they will not show up in the resulting Click-to-Sign signature. To circumvent this, it is necessary to upload a font that contains all the required characters.
• **The text font for Click-to-Sign signatures**
cirrus.document.signing.c2s.font.text
Click-to-Sign signatures need a "text font" that is used to render the text data in a Click-to-Sign signature. If the uploaded data is not usable it will be rejected.

• **The cover page of the final document package**
cirrus.document.signing.final-package.cover-document
The cover page of the final document package can be customized by uploading a flat PDF document. If the uploaded data is not usable it will be rejected.

• **Maximum size of supplemental documents**
cirrus.document.signing.supplemental.file.max-size
The maximum size (in kilobytes) of a single supplemental documents that can be uploaded by a signer.

• **Accepted supplemental documents file suffixes**
cirrus.document.signing.supplemental.file.suffixes
List of accepted file type extensions (file name suffix after period) for supplemental documents. Listed file type extensions must be separated by "," and should not include any whitespace. By default the following file types are accepted: jpg,jpeg,png,doc,docx,pdf.

• **The screen of the signature pad**
cirrus.document.signing.tablet.screen
Defines the layout and content of the screen when signing with a signature pad. If the uploaded data is not usable it will be rejected.

• **Delete audit trail together with package**
cirrus.package.delete.audittrail
Defines whether all package related audit trail entries are deleted automatically if the package is removed.

**Security**

This section lists the security related settings.

• **Signer blocked time**
security.fail.accesscode.blocked.time
Period of time in milliseconds a signer user is blocked before next 2-factor authentication attempt is allowed. This is effective for signers that were blocked after unsuccessful authentication attempts.

• **Maximum number of failed signer authentication attempts**
security.fail.accesscode.max.attempts
Maximum number of failed signer 2-factor authentication attempts allowed until the signer is blocked (see security.fail.accesscode.blocked.time).

• **Reset duration after unsuccessful signer access code authentication**
security.fail.accesscode.max.life.seconds
Maximum number of seconds before the counter for unsuccessful 2-factor authentication attempts for a signer is reset (as long as the signer blocked). The max life time is counted after first failed authentication with a wrong access code. If further authentication attempts fail within the max life time, the signer is blocked (see security.fail.accesscode.blocked.time).
• **Action after too many failed user authentication trials**
  security.fail.login.action
  Action to be taken if maximum number of failed user authentications is reached, supported values are SUSPEND and BLOCK. SUSPEND will suspend the user, only an administrator can activate the user again. BLOCK will block any authentication attempts. The period of blocked time is set by configuration parameter security.fail.login.blocked.time. A reactivation by an administrator is not possible during this time period. This is also effective for users that could not be suspended. A user cannot be suspended if he is the last user with role ADMIN for an account or if the affected user is the last SUPERUSER in the system.

• **User blocked time**
  security.fail.login.blocked.time
  Period of time in milliseconds a user is blocked before next authentication attempt is allowed. This is effective only for users that were blocked after unsuccessful login attempts.

• **Maximum number of failed login attempts**
  security.fail.login.max.attempts
  Maximum number of failed login attempts allowed until a user is suspended or blocked (see security.fail.login.action).

• **Reset duration after unsuccessful user authentication**
  security.fail.login.max.life.seconds
  Maximum number of seconds before the counter for unsuccessful attempts for a user is reset (as long as the user is not suspended or blocked). The max life time is counted after first failed authentication. If further authentication attempts fail within the max life time, the user is suspended or blocked (see security.fail.login.action).

**Mail**

This section lists the mail related settings.

• **Default communication language**
  cirrus.communication.locale
  Default communication language which is used for email notifications. It must be a valid IETF BCP 47 language tag, see https://tools.ietf.org/html/bcp47. Example: en for English or pt-BR for Brazilian Portuguese.

• **List of configurable languages**
  cirrus.config.locales
  The comma separated list of locale strings for storing and retrieving the locale-specific configuration values, like email subject and body. A locale string must be a valid IETF BCP 47 language tag, e.g. en-US, see https://tools.ietf.org/html/bcp47.
• **Final document notification**
  cirrus.mail.finaldocument.notification.enabled
  
  Description
  Currently a Final Document is sent automatically to all signers with an email address after completion of a package.
  The Final Document contains all signed documents and audit trails of a package.
  The default of this account-specific setting cirrus.mail.finaldocument.notification.enabled=[true|false] is true.
  The roles ADMIN and SUPERUSER have read/write access to this setting.
  If the setting is set to false, no email notification is sent to any of the included recipients if a package is completed.
  There are 2 locations for the setting that control the behaviour:
  [Configuration Database] cirrus.mail.finaldocument.notification.enabled=[true|false]
  [cirrus.properties file] cirrus.mail.finaldocument.notification.enabled=[true|false]
  **Important** A configuration setting in the configuration database takes precedence over a setting in cirrus.properties file.

• **Send a mail if a user password has been changed**
  mail.enabled.password.changed
  Send a notification mail to the user if his password has been changed.

• **Access code text for missing delivery plugin**
  mail.message.accesscode.error
  The access code text for a missing or removed delivery plugin. No placeholder expansion is performed here, since this is a placeholder content.

• **Access code text for manual delivery**
  mail.message.accesscode.manual
  The access code text for a manual access code delivery. No placeholder expansion is performed here, since this is a placeholder content.

• **Access code text for delivery by plugin**
  mail.message.accesscode.plugin
  The access code text for delivery by plugin. The only placeholder allowed is % %NOTIFICATIONPLUGINTYPE%% which will query the plugin type used.

• **Account disabled email body**
  mail.message.account.disabled.body
  The body for account disabled emails.

• **Account disabled email subject**
  mail.message.account.disabled.subject
  The subject line for account disabled emails.

• **User invitation email body**
  mail.message.account.invited.body
  The body for user invitation emails.

• **User invitation email subject**
  mail.message.account.invited.subject
  The subject line for user invitation emails.
• **Changed password email body**
  mail.message.changed.password.body
  The body for changed password emails.

• **Changed password email subject**
  mail.message.changed.password.subject
  The subject line for changed password emails.

• **Document copy email body**
  mail.message.email.me.a.copy.body
  The body for document copy emails.

• **Document copy email subject**
  mail.message.email.me.a.copy.subject
  The subject line for document copy emails.

• **Password forgotten email body**
  mail.message.forgotten.password.body
  The body for password forgotten emails.

• **Password forgotten email subject**
  mail.message.forgotten.password.subject
  The subject line for password forgotten emails.

• **Reviewer complete email body**
  mail.message.inform.owner.about.reviewer.complete.body
  The body for owner email after reviewer complete.

• **Reviewer complete email subject**
  mail.message.inform.owner.about.reviewer.complete.subject
  The subject line for owner email after reviewer complete.

• **Signer complete email body**
  mail.message.inform.owner.about.signer.complete.body
  The body for owner email after signer complete.

• **Signer complete email subject**
  mail.message.inform.owner.about.signer.complete.subject
  The subject line for owner email after signer complete.

• **Package cancellation email body**
  mail.message.inform.recipient.about.package.cancellation.body
  The body for recipients email after package cancellation.

• **Package cancellation email subject**
  mail.message.inform.recipient.about.package.cancellation.subject
  The subject line for recipients email after package cancellation.

• **Package expiration email body**
  mail.message.inform.recipient.about.package.expiration.body
  The body for recipients email after package expiration.
• **Package expiration email subject**
  mail.message.inform.recipient.about.package.expiration.subject
  The subject line for recipients email after package expiration.

• **Package complete owner email body**
  mail.message.package.complete.owner.body
  The body for owner email after package complete.

• **Package complete owner email subject**
  mail.message.package.complete.owner.subject
  The subject line for owner email after package complete.

• **Package complete recipient email body**
  mail.message.package.complete.recipient.body
  The body for recipient email after package complete.

• **Package complete recipient email subject**
  mail.message.package.complete.recipient.subject
  The subject line for recipients email after package complete.

• **Decline reason text R1 (documents problem)**
  mail.message.reason.R1
  The decline reason text for R1 (documents problem). No placeholder expansion is performed here, since this is a placeholder content.

• **Decline reason text R2 (sender not recognized)**
  mail.message.reason.R2
  The decline reason text for R2 (sender not recognized). No placeholder expansion is performed here, since this is a placeholder content.

• **Decline reason text R3 (no online signing)**
  mail.message.reason.R3
  The decline reason text for R3 (no online signing). No placeholder expansion is performed here, since this is a placeholder content.

• **Decline reason text R4 (unacceptable terms)**
  mail.message.reason.R4
  The decline reason text for R4 (unacceptable terms). No placeholder expansion is performed here, since this is a placeholder content.

• **Decline reason text R5 (unacceptable terms of GDPR statement)**
  mail.message.reason.R5
  The decline reason text for R5 (unacceptable terms of GDPR statement). No placeholder expansion is performed here, since this is a placeholder content.

• **Signer declined email body**
  mail.message.rejected.body
  The body for signer declined emails.

• **Signer declined email subject**
  mail.message.rejected.subject
  The subject line for signer declined emails.
• **Password reset email body**
  mail.message.reset.password.body
  The body for password reset emails.

• **Password reset email subject**
  mail.message.reset.password.subject
  The subject line for password reset emails.

• **Reviewer notification email body**
  mail.message.reviewing.body
  The body for reviewer notification emails.

• **Reviewer notification email subject**
  mail.message.reviewing.subject
  The subject for reviewer notification emails.

• **Custom reminder email body**
  mail.message.send.message
  The body for custom reminder emails.

• **Custom reminder with link email body**
  mail.message.send.message.with.link
  The body for custom reminder with link emails.

• **Signer notification email body**
  mail.message.signing.body
  The body for signer notification emails.

• **Signer notification email subject**
  mail.message.signing.subject
  The subject for signer notification emails.

• **Default value for signer notification email subject**
  mail.message.signing.subject.default
  The default subject text for package specific signer notification emails. When necessary, this string can contain special characters that are replaced with meaningful data at runtime. Available special characters are $USER which is replaced by the current user name and $NOW which is replaced by the current time.

• **Default value for signer notification email text**
  mail.message.signing.text.default
  The default value for the package specific message in signer notification emails. When necessary, this string can contain special characters that are replaced with meaningful data at runtime. Available special characters are $USER which is replaced by the current user name and $NOW which is replaced by the current time.

• **Team invitation email body**
  mail.message.user.add.to.team.invited.body
  The body for team invitation emails.

• **Team invitation email subject**
  mail.message.user.add.to.team.invited.subject
  The subject line for team invitation emails.
Plugins

This section lists the general settings related to plugins.

Enabled

This section contains the enabled plugins.

Configuration

In this section the enabled plugins can be configured.

General

• Plugin directory
  plugin.directory
  The directory where plugins will be located (in addition to the CLASSPATH).

• Plugin load list
  plugin.loadlist
  The list of plugin ids to be loaded. Ids must be separated by ‘,’.

Client

This section lists the client related settings.

Signing Client related settings

• Requirement for e-sign consent
  client.signing.esign.consent.required
  The requirement for displaying the e-sign consent text which must be agreed by a recipient before signing or reviewing a signing package.

• E-sign consent text
  client.signing.esign.consent.text
  The e-sign consent text which must be agreed by a recipient before signing or reviewing a signing package.

• The external e-sign consent URL
  client.signing.esign.consent.url
  The custom e-sign consent URL which is provided as link in the Signing Client in addition the e-sign consent text (max 2000 chars).

• Requirement for GDPR consent
  client.signing.gdpr.required
  The requirement for displaying the GDPR (EU General Data Protection Regulation) consent text which must be agreed by a recipient before signing or reviewing a signing package.

• GDPR statement
  client.signing.gdpr.text
  GDPR (EU General Data Protection Regulation) text which must be agreed by a recipient before signing or reviewing a signing package.
• The external GDPR policy URL
  client.signing.gdpr.url
  The custom GDPR (EU General Data Protection Regulation) URL which is provided as link in the
  Signing Client in addition the GDPR data protection statement (max 2000 chars).
• The external finish URL
  client.signing.view.finish.url
  The custom URL which is called when a signing session is finished by the remote signer. If no URL is
  provided the default finish page is displayed in the signing-client.
• Show decline action
  client.signing.view.general.decline.visible
  Defines if the decline action is visible in the signing-client. If the action is not visible the signer is not
  able to decline a signing session.
• Show footer
  client.signing.view.general.footer.visible
  Defines if the footer is shown in the signing client.
• Show header
  client.signing.view.general.header.visible
  Defines if the header is shown in the signing client.
• Show instructions
  client.signing.view.general.instructions.visible
  Defines if the instructions are shown in the signing client.
• Show wizard-steps
  client.signing.view.general.wizardsteps.visible
  Defines if the wizard-steps are shown in the signing client.
• Show "In-person Signing" view
  client.signing.view.inperson.visible
  Defines if the in-person signing view is shown in the signing client. If set to false the in-person signing
  view is skipped when there is only one signer in an in-person signing session.
• Show download in "Review & Sign" view
  client.signing.view.reviewsign.download.visible
  Defines if the download actions are shown in the review & sign view.
• Show progress in "Review & Sign" view
  client.signing.view.reviewsign.progress.visible
  Defines if the progress bar is shown in the review & sign view.
• Show "RESUME LATER" action in "Review & Sign" view
  client.signing.view.reviewsign.resume_later.visible
  Defines if the resume later action is shown in the review & sign view.
• Show "Review & Sign" view
  client.signing.view.reviewsign.visible
  Defines if the review & sign view is shown in the signing client. If set to false the review & sign view
  is skipped when only one document is used in the signing session and no view-specific features are
  assigned to the signer, like TSP and supplemental documents.
• Show "Welcome" view
client.signing.view.welcome.visible
Defines if the welcome view is shown in the signing client. If set to false the welcome view is skipped when no signer authentication (access code or external authentication) is used.

Manage Client related settings

• Signer names client colors
client.general.signer.colors
The client displays each signer in a specific color. The setting contains a list of hexadecimal RGB colors (#RRGGBB) separated by commas. If there are more signers than colors, the colors are repeated.

• Recipients must be selected
client.manage.restrict.recipients.input
If true, recipients cannot be entered manually, but must be selected from a list.

• Takeover attributes of selected signer
client.manage.signer.autocomplete.takeover
Signer attributes are adopted from the selected signer after autocomplete search.

• Generate first and last name proposal
client.manage.signer.name.proposal
Defines whether the client fills the first name and last name fields with proposals derived from the signer name if external authentication is selected as authentication method for a signer.

Advanced signing settings

This section lists the advanced settings related to the signing process.

• 2FA access code length
cirrus.security.2fa.accesscode.length
The length of the generated random access code for the two factor authentication.

• The external authentication provider name
cirrus.security.external.authentication.name
The external authentication provider name. This name is displayed in the Manage Client as well as in the Signing Client as identification for the implemented external authentication service.

• The external authentication service shared secret
cirrus.security.external.authentication.sharedsecret
The external authentication service shared secret used to authenticate the system REST calls.

• The external authentication service URL
cirrus.security.external.authentication.url
The custom authentication service application URL.

• Variable part of the application configuration shared secret
cirrus.security.ksd_appconf_shasec
The application configuration shared secret is used to encrypt client to server communication. This setting lets you personalize the encryption.
• **Signature device selection priority when signing**
  client.signing.device.priority
  Defines the signature device selection priority (i.e. the SignWare padclass attribute) when signing. (SPTabletWSignPad, SPTabletWTablet, SPTabletWSignPad;SPTabletWTablet, SPTabletWTablet;SPTabletWSignPad) Values must be entered without whitespaces.

• **Signature image maximum size**
  client.signing.signature.image.size.max
  The maximum size (in KB) of an uploaded signature image which can be used for signing. The allowed range is between 1 and 1000 (KB).

• **Signature (certificate) type**
  client.signing.tsp.signature.type
  Signature (certificate) type which is needed for signing (BASIC, ADVANCED or QUALIFIED).
Chapter 4

Plugins

Plugins are SignDoc Standard extensions that can be created by the customer to extend or customize SignDoc Standard functionality. The plugin interfaces are described in the SignDoc Standard Developer's Guide, see Related documentation.

Plugins can be enabled and configured on an account basis. Any account can have its own plugins enabled. Plugins can be configured differently for different accounts.

Enable plugins

Plugins are enabled on two levels:

• Plugin load list
  The plugin load list controls which plugins are available in the SignDoc Standard application.

• Plugin enabled
  This setting enables a specific plugin globally, or on an account basis.

To use a plugin by a specific account, the plugin has to be added to the load list, then once it is available, it has to be enabled and configured for that specific account.

Plugin load list

The plugin load list is a comma separated list of plugin class names. All plugins listed in the list are loaded by SignDoc Standard and thus made available for use. The load list can only be edited by a server administrator. Thus, the server administrator maintains control of the plugins used by the application.

To add a plugin to the load list, the server administrator can use the Administration Center and edit the global load list setting. The load list is not available on an account basis. It affects the whole system.

Alternatively, the server administrator can also use the REST API and set the load list as any other configuration setting.

After the load list has been changed, the application will reload all plugins and make the settings of the new plugins available. No system restart is required.

Plugin enabled

Once a plugin has been loaded by adding it to the load list, its configuration settings will be made available for configuration. One of the settings is the 'enabled' switch. It has to be turned to 'true' to activate a plugin. It also controls how the plugin will be run.
Plugins can be activated:
• On a global basis for all accounts.
  This is accomplished by activating the ‘enabled’ global setting for the plugin.
• On an account basis.
  The ‘enabled’ setting can be set in the account-specific configuration.

It is recommended to enable and configure a plugin during the same editing step, then clicking SAVE, since when a plugin is enabled, the system will actually attempt to initialize it using its configuration options.

Plugins can be activated by:
• The server administrator.
  A server administrator can enable a plugin both globally (for the whole system) and for a specific account. The server administrator is able to activate a plugin for any account on the system.
• The account administrator.
  An account administrator can activate plugins only for the one account he is administering.

Configure plugins

Each plugin will have a different list of configuration options. A plugin will register the configuration options it needs with the SignDoc Standard application. The options can then be configured using the configuration GUI in the Administration Center or Manage Client, or using the REST API.

Plugin configuration options can be set the same way a plugin is enabled:
• On a global basis for all accounts.
  These settings can only be configured by the server administrator in the Administration Center.
• On an account basis.
  These settings can be set either by the server administrator in the Administration Center, or by the account administrator in the Manage Client.

Sample workflow to enable a plugin

This section describes the different ways to enable a plugin using the example of the ClamAV virus scan plugin.

To actually be able to use the plugin you have to set up a ClamAV server described in the section ClamAV virus scan plugin and to provide the scan URL to the document scan plugin.

Enable a plugin globally

You have to log in as a server administrator to be able to load and enable a plugin globally.

Add the plugin to be configured globally to the load list

The ClamAV class name is ‘de.softpro.cirrus.plugins.document_scan.ScanClamAV’.
Open the Administration Center, log in as a server administrator and click the **System settings** link.

Click the **General** section below the Plugins category and change the **Plugin load list** setting by adding the class name to the comma delimited list:

After adding the name to the list, save the setting by clicking the **SAVE** button. Give the application 10 – 30 seconds to reload the plugin list. You will have to refresh the Administration Center to show the new settings.

**Enable and configure the plugin globally**

After you have refreshed the client, it will show two new settings:

- **Plugin enabled: ScanClamAV** in the **Enabled** section
- **plugin.cfg.ScanClamAV.url** in the **Configuration** section

Slide the **Plugin enabled: ScanClamAV** setting to on:
Set the URL to point to the URL of the ClamAV server you are running by editing the URL setting in the *Configuration* section:

Save the settings by clicking the **SAVE** button.
The plugins will be reloaded and an instance of the ClamAV document scanning plugin should now be running, available for all accounts on the system.

**Enable a plugin for a specific account**

If only one specific account should have the ClamAV scanner enabled, you will have to configure the plugin for that specific account.

**Add the plugin to be configured for a specific account to the load list**

This step does not differ from the one shown above. It has to be performed by a server administrator.

Open the Administration Center, log in as a server administrator and then click the **System settings** link.

Click the **General** section below the Plugins category and change the **Plugin load list** setting by adding the class name to the comma delimited list:

![Administration Center screenshot](image)

After adding the name to the list, save the setting by clicking the **SAVE** button. Give the application 10 – 30 seconds to reload the plugin list. You will have to refresh the Administration Center to show the new settings.

**Enable and configure the plugin for a specific account**
Enabling the plugin for a specific account can be done by either the server administrator or an account administrator.

The server administrator will use the Administration Center, similar to the example for Enable a plugin globally, but will edit an individual account setting instead of the global settings for all accounts.

The account administrator can use the Manage Client to perform the same operation.

Log into the Manage Client with account administrator credentials.

Go to the Administration section and find Plugins, Configuration. Find the ClamAV URL setting and point it to the ClamAV server you are running. Save the setting.

Change to the Enabled section and enable the plugin by sliding the switch to on, then save the setting.

The plugin will be loaded and will be available for the account you configured only.

To make the plugin available for a different account you will have to log in as an account administrator for the other account and repeat the process. You can use different settings for different accounts. In case of the ClamAV plugin, different accounts can use different ClamAV scanning servers.
Core plugins

This section contains a list of plugins that are provided with the SignDoc Standard product. See the section Configure plugins on how to configure a specific plugin.

- ClamAV virus scan plugin
- KTA state change plugin
- TSP plugin
- SMS notification plugin

ClamAV virus scan plugin

- Document scan event
- ClamAV virus scanner
- Configure the ClamAV document scan plugin
- Test the scanning

Document scan event

To prevent invalid documents from being uploaded or viruses being spread to customers via uploaded infected documents, SignDoc Standard supports the document scan event interface.

Whenever a document or supplemental document is being uploaded, a document scan event is posted to all registered plugins on the affected account. All plugins can scan the document content. If any plugin responds with ‘true’ as to invalid content being detected, the document upload is rejected.

A sample implementation of this plugin for the ClamAV virus scanner is provided. The customer can implement any other scan implementation of his choice, if a different scanner is required.

ClamAV virus scanner

ClamAV

For the sample document scan implementation the ClamAV virus scanner has been used. The main reasons for this choice are:

- ClamAV is an open source virus scanner
- Is available free of charge
- Available on all major operating systems

Information on ClamAV can be found under https://www.clamav.net

Kofax does not provide support on installing or running a ClamAV server.

Running the ClamAV scan server

To scan documents via the sample ClamAV document scan plugin a ClamAV server and a ClamAV REST endpoint need to be running.

- Documentation on running a ClamAV server can be found under https://www.clamav.net
The REST endpoint is documented under https://github.com/solita/clamav-rest

The easiest way to run this combination is using a Docker Compose setup (docker-compose.yml):

```yaml
version: '3'
services:
  clamav:
    image: mkodockx/docker-clamav
    ports:
    - "3310:3310"
  clamav-rest:
    image: lokori/clamav-rest
    links:
    - clamav
    ports:
    - "8080:8080"
    environment:
    - CLAMD_HOST=clamav
```

The above docker-compose.yml shows how to start two Docker containers:
- clamav runs a normal ClamAV server
- clamav-rest runs the REST endpoint connected to clamav server

The compose setup can be started using `docker-compose up -d`. The URL endpoint for this setup will be http://hostname:8080/scan.

Kofax does not provide support on installing or running the ClamAV server. Refer to the links above and to https://www.docker.com for additional information.

Configure the ClamAV document scan plugin

Plugin load list

The load list specifies which plugins are supported by Cirrus. To make the ClamAV document scan plugin usable, it has to be added to the load list.

The load list is a ',' delimited list of plugin class names.

```text
plugin.loadlist = de.softpro.cirrus.plugins.document_scan.ScanClamAV
```

Enabling the plugin

Once the plugin has been loaded it can be enabled:
- For one or more specific accounts individually
- For all accounts globally

To enable the plugin you have to change the setting

```text
plugin.enabled.ScanClamAV = true
```

either globally (no account id), or for a specific account.

Plugin settings
Plugin settings are set as

PLUGIN.CFG.<PLUGINID>.<SETTINGNAME>

Example

plugin.cfg.ScanClamAV.url

Following settings are supported by the ScanClamAV plugin:

- **url** The URL to the ClamAV server REST endpoint (see above). For the sample server configuration provided it has the form:
  
  http://hostname:8080/scan

Test the scanning

To test virus detection one can upload the test EICAR virus signature as a document and verify that the scanner works ([http://www.eicar.org](http://www.eicar.org)).

**KTA state change plugin**

- **KTA**
- **State change events**
- **Configuration**

**KTA**

This plugin implements the communication with the KTA (Kofax TotalAgility) system. KTA will create signing packages with documents to be signed as part of its workflow. After the documents have been signed, this plugin will report the status change back to the KTA system and let it continue its workflow.

Refer to the KTA documentation on how to configure the KTA system to generate signing packages with SignDoc.

Kofax SignDoc Standard introduces this new KTA state change plugin with version 2.1.0.

**State change events**

The plugin implements the state change event interface and reacts to SignDoc state changes:

- Signing package state changes: Change from any state to CANCELED, REJECTED, EXPIRED or COMPLETE
- Signer state changes: Change from any state to COMPLETE

Upon receiving the relevant state change events, the plugin will generate KTA calls to inform KTA on the new processing state.

**Configuration**

**Plugin load list**

The load list specifies which plugins are supported by Cirrus. To make the KTA state change plugin usable, it has to be added to the load list.
The load list is a ',' delimited list of plugin class names.

```
plugin.loadlist =
de.softpro.cirrus.plugins.state_change.KTAStateChangeNotification
```

**Enabling the plugin**

Once the plugin has been loaded via the load list it can be enabled:

- For one or more specific accounts individually
- For all accounts globally

To enable the plugin, you have to set the setting

```
plugin.enabled.KTAStateChangeNotification = true
```

either globally (no account id), or for a specific account.

**Plugin settings**

Plugin settings are set as

```
plugin.cfg.<pluginId>.<settingName>
```

**Example**

```
plugin.cfg.KTAStateChangeNotification.ktaurl
```

Following settings are supported by the KTA state change notification plugin:

- **ktaurl** The KTA URL to send information to.
- **cirrusurl** The Cirrus (Signdoc Standard) REST API URL. The plugin will use the REST API to obtain additional signing package and signer information needed to process the request. Specify the URL without the REST API version number! Example: http://your.host.name/cirrus/rest
- **sessionid** The KTA session id.
- **jobnotetemplate** Optional. If you need to change the job note template you can set this parameter. This was previously done by providing a job note template in the Cirrus home directory.

**Additional required configuration setting**

If the KTA state change plugin is enabled it is required that the setting

```
cirrus.document.prepare.msword.signatureline.signerid.source = field_name
```

in the category "Documents and packages" is set.

**TSP plugin**

- Digital document signing and trusted service providers
- Registering an account with Bundesdruckerei
- Configuration
Digital document signing and trusted service providers

As of release 1.3.1 Kofax SignDoc supports digitally signing documents according to the EIDAS standard. The document is signed by an external trusted service provider, independent of the Kofax SignDoc installation. The TSP is accessed via a plugin interface, allowing for multiple TSP support and expansion independent of a new product release.

The standard plugin delivered with the product supports DTrust GmbH / Bundesdruckerei GmbH as a trusted service provider. Other providers can be added by writing and configuring additional plugins. This chapter describes the configuration of the plugin mentioned above.

Registering an account with Bundesdruckerei

To use the service, both the operator and the signers need to have an account registered with Bundesdruckerei GmbH.

The operator has to register with Bundesdruckerei GmbH and obtain a partner id and authentication settings.

Any signer that will have a digital signature appended during the signing process has to register with Bundesdruckerei GmbH as a user.

Click [https://cloud-ref.sign-me.de/signature/start](https://cloud-ref.sign-me.de/signature/start) to go to the registration webpage of Bundesdruckerei GmbH.

Configuration

All settings described here are configured via the Cirrus configuration service. No specific SignDoc Web configuration is needed, since the component will request the configuration settings via the Cirrus configuration API.

Plugin load list

```plaintext
plugin.loadlist = de.softpro.cirrus.plugins.tsp.TSPBundesdruckerei
```

The load list specifies which plugins are supported by Cirrus. This is a system-wide setting and can only be set account independent. To load the TSP plugin it has to be added to it, which is a list of ‘,’ delimited class names.

Enabling the plugin

Once a plugin has been loaded via the load list it can be enabled:

for one or more specific accounts

for all accounts globally

To enable the plugin, you have to set the setting

```plaintext
plugin.enabled.TSPBundesdruckerei = true
```

either as a global setting (no account) or for one or more specific account ids.

TSP plugin settings

Plugin settings are set as
plugin.cfg.<pluginId>.<settingName>

Example

plugin.cfg.TSPBundesdruckerei.visibility

They can be set either as global values or on an account-specific basis.

Following settings are supported by the TSPBundesdruckerei plugin:

- `regurl` The registration URL where new Bundesdruckerei GmbH users can register an account.
- `apiurl` SignMe API url.
- `basicauthuid` Basic authentication user id received from Bundesdruckerei GmbH.
- `basicauthpw` Basic authentication password received from Bundesdruckerei GmbH.
- `partnerauthuid` Partner authentication user id received from Bundesdruckerei GmbH.
- `partnerauthpw` Partner authentication password received from Bundesdruckerei GmbH.
- `visibility` Visibility of document signatures. Possible values are INVISIBLE, FIRSTPAGE, ALLPAGES, default being INVISIBLE.
- `padesform` PAdES format. Possible values are BASIC or ENHANCED, default being BASIC. When using ENHANCED the parameter visibility has to be set to FIRSTPAGE or ALLPAGES.
- `replacefilename` Replace file names sent for signing if they contain unsupported characters. Possible values: true or false. If true the file name will be changed for display in SignMe to "document.pdf".

Sample TSP plugin configuration

Enable TSP plugin for Bundesdruckerei via configuration REST API (required role SUPERUSER):

POST /rest/v5/configuration

Add plugin to plugin load list:

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

Enable plugin for specific account (here signdoc):

POST /rest/v5/configuration?accountid=signdoc

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

Provide plugin settings for specific account:

```json
{

}```
SMS notification plugin

- Notification and the SMS plugin
- Registering an account with the SMS service
- Configuration

Notification and the SMS plugin

The notification service is designed to send a message to the user / signer. For a two factor authentication the login information is split into two parts: the login link and the access code.

The access code has to be delivered via a different channel than the login link.

To be able to support different types of delivery channels, the feature is implemented via a plugin. Thus additional delivery channels can be supported without changes to the product.

The core plugin supports an SMS notification channel.

Registering an account with the SMS service

To use the SMS plugin you will have to own a user account with Clickatell (http://www.clickatell.com).
Clickatell changed its account structure and API in November 2016. Therefore two different plugins are provided, depending on the type of account you have registered:

- Accounts registered before November 2016 (Clickatell Central): use the NotificationSMSClickatell plugin.
- Accounts registered after November 2016 (Clickatell Platform): use the NotificationSMSClickatellPlatform plugin.

The plugins can be used at the same time in an installation. Usually one account will only use a single plugin. Both plugins return ‘SMS’ as a delivery channel.

Configuration

All settings described here are configured via the Cirrus configuration service.

Currently the configuration service is reachable via the REST API, or the configuration editors in the Manage Client or Administration Center.

The following sections will describe the configuration for each type of plugin.

Clickatell Central API

Plugin load list

The load list specifies which plugins are supported by Cirrus. To make the SMS notification plugin usable, it has to be added to the load list.

The load list is a ‘,’ delimited list of plugin class names.

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

Enabling the plugin

Once the plugin has been loaded via the load list it can be enabled:

- For one or more specific accounts
- For all accounts globally

To enable the plugin, you have to set the setting

```
plugin.enabled.NotificationSMSClickatell = true
```

either as a global setting (no account) or for a specific account id.

SMS plugin settings

Plugin settings are set as:

```
plugin.cfg.<pluginId>.<settingName>
```

Example

```
plugin.cfg.NotificationSMSClickatell.url
```
Following settings are supported by the NotificationSMSClickatell plugin:

- **url** SMS service URL. The URL where the Clickatell SMS service can be reached. Usually https://api.clickatell.com/http/sendmsg
- **userid** SMS service user id. The user id of the Clickatell SMS service user used to authenticate with the service.
- **password** SMS service password. The password of the Clickatell SMS service user used to authenticate with the service.
- **apiid** SMS service API id. The API ID you received when setting up the user and http service at Clickatell.
- **senderid** Sender id to use (numeric only 16 digits or alphanumeric 11 characters, registered). If you want to use a sender id with the sent SMS messages, you have to register the sender id at Clickatell. After successful registration you can specify the registered sender id here. The sender id can be either a telephone number, or an alphanumeric id. Alphanumeric sender ids are limited to 11 characters in length.
- **utf16** Use UTF-16BE encoding (true / false). The SMS alphabet is limited regarding the characters that can be used for the message. If special characters or locales (Chinese, Japanese, etc.) or symbols will be used in the message, UTF16 encoding can be used. The UTF encoding however limits the message length. Thus it should only be used if necessary.
- **maxparts** Maximum number of message parts to be used (1-3, default 3). SMS messages are limited in length. Longer messages can be sent by chaining SMS parts together (at additional cost). This setting specifies the maximum number of message parts the system will send.
- **userparam** Additional parameters to be sent to the service (use URL encoding). If additional Clickatell settings need to be used they can be specified here. The parameters will have to be URL encoded.

**Testing howto**

The steps needed with the Swagger UI to get a plugin configured are described below:

1. Get access token for ksdadmin (no account). Use:
   users > create an authentication token
2. Set plugin load list account independent. Use:
   configuration > set configuration settings
   
   ```
   [ 
   { 
   "k": "plugin.loadlist",
   "v": "de.softpro.cirrus.plugins.notification.NotificationSMSClickatell"
   }
   ]
   ```
3. Set plugin cfg (can be account specific). Use:
   configuration > set configuration settings
   
   ```
   [ 
   { 
   "k": "plugin.enabled.NotificationSMSClickatell",
   "v": "true"
   },
   { 
   "k": "plugin.cfg.NotificationSMSClickatell.url",
   "v": "http://smscatcher.sdlabs.de:8080/sendmsg"
   },
   { 
   "k": "plugin.cfg.NotificationSMSClickatell.userid",
   "v": "47"
   }
   ```
Clickatell Platform API

Plugin load list

The load list specifies which plugins are supported by Cirrus. To make the SMS notification plugin usable, it has to be added to the load list.

The load list is a ',' delimited list of plugin class names.

```plaintext
plugin.loadlist =
de.softp.pro.cirrus.plugins.notification.NotificationSMSClickatellPlatform
```

Enabling the plugin

Once the plugin has been loaded via the load list it can be enabled:
- For one or more specific accounts
- For all accounts globally

To enable the plugin, you have to set the setting

```plaintext
plugin.enabled.NotificationSMSClickatellPlatform = true
```

either as a global setting (no account) or for a specific account id.

SMS plugin settings

Plugin settings are set as:

```plaintext
plugin.cfg.<pluginId>.<settingName>
```

Example

```plaintext
plugin.cfg.NotificationSMSClickatellPlatform.url
```

Following settings are supported by the NotificationSMSClickatellPlatform plugin:
- **url** SMS service URL The URL where the Clickatell SMS service can be reached. Usually [https://platform.clickatell.com/messages/http/send](https://platform.clickatell.com/messages/http/send)
- **apikey** SMS service API key. The Clickatell Platform API key you have set up by creating a new http integration on your Clickatell account.
- **senderid** Optional sender id to use (numeric only 16 digits or alphanumeric 11 characters, registered).
  If you want to use a sender id with the sent SMS messages, you have to register the sender id at Clickatell. After successful registration you can specify the registered sender id here. The sender id
can be either a telephone number, or an alphanumeric id. Alphanumeric sender ids are limited to 11 characters in length.

Testing howto

The steps needed with the Swagger UI to get a plugin configured are described below:

1. Get access token for ksadmin (no account). Use:
   users > authentication

2. Set plugin load list account independent. Use:
   configuration > set configuration settings

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

3. Set plugin cfg (can be account specific). Use:
   configuration > set configuration settings

   ```json
   [
   {
     "k": "plugin.enabled.NotificationSMSClickatellPlatform",
     "v": "true"
   },
   {
     "k": "plugin.cfg.NotificationSMSClickatellPlatform.url",
     "v": "https://platform.clickatell.com/messages/http/send"
   },
   {
     "k": "plugin.cfg.NotificationSMSClickatellPlatform.apikey",
     "v": "Your-API-key=="
   }
   ]
   ```
Chapter 5

Monitoring

Overview

This chapter describes the current implementation of SignDoc monitoring. The metrics and protocol provided by the SignDoc implementation are shown. In addition, a sample monitoring setup using standard components (Collectd, InfluxDB, Grafana) is described.

The main design goals are:

• Providing key application metrics:
  • In addition to the usual system monitoring, key application metrics are recorded to be able to track application performance in production and detect problems in a cloud environment.

• Using well established protocols:
  • The Graphite protocol is widely used by applications to provide metrics to monitoring systems.
  • In addition to the sample monitoring server described in this document, interfaces are available to use the protocol on most monitoring systems, including DataDog, currently used by Kofax cloud operations.

• Using standard and highly configurable components for the sample setup:
  • All components used have an established track record in application monitoring and are available at no additional cost.

• Minimizing resource usage:
  • Metrics and naming are designed to minimize resource (notably network) usage. Metrics reporting can be filtered to exclude unwanted information.
Monitoring the KSD cluster

Monitoring protocol

Metrics are provided via the plaintext Graphite protocol.

Metrics are sent in the form:

<metric path> <metric value> <metric timestamp>

where metric path is a ‘.’ separated list of names defining the metric.

One sample metric could look like:

local.random.diceroll 4 1445620537

Data is sent by default via UDP, TCP can be configured alternatively if needed.

Each application sends the metrics at configured time intervals to the server recording the data.
Metrics being collected

Metrics can be collected on different aspects of the cluster operation:

- The computer (or VM) the application runs on
- Docker container metrics (by container)
- Java JMX metrics
- Application metrics

Application metrics and selected JMX metrics are collected by the KSD application and reported to the monitoring system.

This document also describes how system and Docker container metrics can be collected with the collectd container.

Monitoring server

Metrics can be sent to any monitoring system supporting the Graphite plaintext protocol (TCP or UDP).

This document also describes a monitoring setup using InfluxDB as a data store and Grafana for dashboard creation. This is a sample setup, not part of the product deliverable or support.

SignDoc Standard application metrics
Metric overview

Metrics are being collected for:

• Application: metrics describing the application state.
• Java JMX: selected metrics describing the Java state (memory, threads, etc.).
• Tomcat metrics via JMX: selected metrics describing the tomcat handler state.
• Docker container metrics: describing individual docker containers on a machine (cpu, memory, network, etc.).
• System metrics: describing the state of the machine the application is being run on (cpu usage, memory, disk, network i/o, etc.).

For all metrics the following description applies.

All metric names start with <instance>.<profile> where:

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>instance</td>
<td>The instance, or hostname, where the data is being collected.</td>
</tr>
</tbody>
</table>
| profile | The profile of the data.  
| | • In case of system metrics, this is the category of the data (cpu, memory, disk, etc.).  
| | • In case of docker containers it is the container id or name.  
| | • In case of application metrics it is the assigned container name or id. |

Metrics reporting statistical information have following endings:

<table>
<thead>
<tr>
<th>Ending</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>count</td>
<td>The count of events.</td>
</tr>
<tr>
<td>m1_rate</td>
<td>One minute exponentially weighted moving average rate of event occurrence.</td>
</tr>
<tr>
<td>m5_rate</td>
<td>Five minute exponentially weighted moving average rate of event occurrence.</td>
</tr>
<tr>
<td>m15_rate</td>
<td>Fifteen minute exponentially weighted moving average rate of event occurrence.</td>
</tr>
<tr>
<td>mean_rate</td>
<td>Mean rate of event occurrence.</td>
</tr>
<tr>
<td>p50</td>
<td>The value of the 50th percentile of the distribution.</td>
</tr>
<tr>
<td>p75</td>
<td>The value of the 75th percentile of the distribution.</td>
</tr>
<tr>
<td>p95</td>
<td>The value of the 95th percentile of the distribution.</td>
</tr>
<tr>
<td>p98</td>
<td>The value of the 98th percentile of the distribution.</td>
</tr>
<tr>
<td>p99</td>
<td>The value of the 99th percentile of the distribution.</td>
</tr>
<tr>
<td>p999</td>
<td>The value of the 999th percentile of the distribution.</td>
</tr>
<tr>
<td>stddev</td>
<td>The standard deviation of the distribution.</td>
</tr>
<tr>
<td>min</td>
<td>The minimum value of the distribution.</td>
</tr>
<tr>
<td>max</td>
<td>The maximum value of the distribution.</td>
</tr>
<tr>
<td>mean</td>
<td>The mean value of the distribution.</td>
</tr>
</tbody>
</table>
Statistical values are usually recorded in an exponentially weighted moving window. Recent events have a higher impact on the statistical values.

## Configuration

**Important** SignDoc Standard before version 2.1.0 was mainly configured with the configuration file cirrus.properties. This file moved to `INSTALLDIR\_conf_templates\cirrus.properties` with version 2.1.0.

Since SignDoc Standard 2.1.0, it is highly recommended to use the file `INSTALLDIR\service_configuration.properties` (instead of cirrus.properties) whenever it is required to configure SignDoc with a configuration file. Configurations set in this file are applied as Java System Property and have therefore highest precedence.

Starting with the release 1.3, some configuration options have been moved from configuration files to the configuration service.

Monitoring and the metrics processed can be configured in cirrus.properties configuration file with the following properties:

- **monitoring.host** The hostname or IP address of the monitoring server the metrics are sent to. If left empty, no monitoring data is sent (default).
- **monitoring.port** The port number where monitoring data is sent to. Defaults to 2003
- **monitoring.protocol** The protocol being used (TCP or UDP). Defaults to UDP.
- **monitoring.filter.include** Regular expression to specify which metrics form the available list should be reported. Defaults to all.
- **monitoring.filter.exclude** Regular expression specifying which metrics should be excluded form monitoring. By default, following statistic information is being excluded: count, m5_rate, m15_rate, max, min, mean_rate, p50, p75, p95, p98, p99, p999, stddev

### Metric description

#### Requests

HTTP request data for the two types of requests processed: cirrus and rest:

- `<instance>.<profile>.cirrus.req.<req_type>.count`
- `<instance>.<profile>.cirrus.req.<req_type>.m1_rate`
- `<instance>.<profile>.cirrus.req.<req_type>.m5_rate`
- `<instance>.<profile>.cirrus.req.<req_type>.m15_rate`
- `<instance>.<profile>.cirrus.req.<req_type>.mean_rate`
- `<instance>.<profile>.cirrus.req.<req_type>.p50`
- `<instance>.<profile>.cirrus.req.<req_type>.p75`
<instance>.<profile>.cirrus.req.<req_type>.p95
<instance>.<profile>.cirrus.req.<req_type>.p98
<instance>.<profile>.cirrus.req.<req_type>.p99
<instance>.<profile>.cirrus.req.<req_type>.p999
<instance>.<profile>.cirrus.req.<req_type>.stddev
<instance>.<profile>.cirrus.req.<req_type>.mean
<instance>.<profile>.cirrus.req.<req_type>.min
<instance>.<profile>.cirrus.req.<req_type>.max

Response status

HTTP response statistics by status code.

<instance>.<profile>.cirrus.resp.<resp_type>.<resp_status>.count
<instance>.<profile>.cirrus.resp.<resp_type>.<resp_status>.m1_rate
<instance>.<profile>.cirrus.resp.<resp_type>.<resp_status>.m5_rate
<instance>.<profile>.cirrus.resp.<resp_type>.<resp_status>.m15_rate
<instance>.<profile>.cirrus.resp.<resp_type>.<resp_status>.mean_rate

Where:

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>resp_type</td>
<td>The response type. Currently rest and cirrus.</td>
</tr>
<tr>
<td>resp_status</td>
<td>The group of status codes:</td>
</tr>
<tr>
<td></td>
<td>httpStatus2XX: 200 – 299 status codes</td>
</tr>
<tr>
<td></td>
<td>httpStatus3XX: 300 – 399 status codes</td>
</tr>
<tr>
<td></td>
<td>httpStatus4XX: 400 – 499 status codes</td>
</tr>
<tr>
<td></td>
<td>httpStatus5XX: 500 – 599 status codes</td>
</tr>
<tr>
<td></td>
<td>httpStatusXXX: all other status codes</td>
</tr>
</tbody>
</table>

REST API calls

Metrics describing the number and duration of REST API calls.

<instance>.<profile>.rest.<ver>.<controller>.<function>.count
<instance>.<profile>.rest.<ver>.<controller>.<function>.m1_rate
<instance>.<profile>.rest.<ver>.<controller>.<function>.m5_rate
<instance>.<profile>.rest.<ver>.<controller>.<function>.m15_rate
For each API call following fields are set:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ver</td>
<td>The REST API version. Currently v2 and v3.</td>
</tr>
</tbody>
</table>
| controller | The REST API controller:  
a: account controller  
d: document controller  
p: signing package controller  
r: reminder controller  
s: signer controller (currently unused)  
t: team controller  
u: user controller  
y: system controller  
| function | The function being called. |

For the REST API v2, only the createPackage function is being monitored. For the REST API v3 all API functions are being monitored.

**Queues**

Selected data on the internal ActiveMQ message queues used by the application: task and notification.

- Average amount of time, in milliseconds, that messages sat in the queue before being consumed:
  \[<\text{instance}>,\text{<profile>},\text{cirrus.queue.<queue_type>}.averageEnqueueTime\]

- Number of messages that have been acknowledged and removed from the queue:
  \[<\text{instance}>,\text{<profile>},\text{cirrus.queue.<queue_type>}.dequeueCount\]

- Percentage of available memory in use:
  \[<\text{instance}>,\text{<profile>},\text{cirrus.queue.<queue_type>}.memoryPercentUsage\]
JMX metrics

This section describes the selection of Java JMX metrics made available by the application. For the description refer to the Java JMX documentation.

Memory

<instance>.<profile>.jmx.memory.heap.committed
<instance>.<profile>.jmx.memory.heap.init
<instance>.<profile>.jmx.memory.heap.max
<instance>.<profile>.jmx.memory.heap.usage
<instance>.<profile>.jmx.memory.heap.used
<instance>.<profile>.jmx.memory.non-heap.committed
<instance>.<profile>.jmx.memory.non-heap.init
<instance>.<profile>.jmx.memory.non-heap.max
<instance>.<profile>.jmx.memory.non-heap.usage
<instance>.<profile>.jmx.memory.non-heap.used
<instance>.<profile>.jmx.memory.total.committed
<instance>.<profile>.jmx.memory.total.init
<instance>.<profile>.jmx.memory.total.max
<instance>.<profile>.jmx.memory.total.usage
<instance>.<profile>.jmx.memory.total.used

Threads

<instance>.<profile>.jmx.thread.daemonThreadCount
<instance>.<profile>.jmx.thread.peakThreadCount
<instance>.<profile>.jmx.thread.threadCount
<instance>.<profile>.jmx.thread.totalStartedThreadCount

Garbage collection

<instance>.<profile>.jmx.gc.MarkSweep.collectionCount
<instance>.<profile>.jmx.gc.MarkSweep.collectionTime
<instance>.<profile>.jmx.gc.Scavenge.collectionCount
<instance>.<profile>.jmx.gc.Scavenge.collectionTime
Tomcat processor metrics

A selection of the Tomcat handler (global processor) metrics are made available, for each of the handlers configured.

<instance>.<profile>.catalina.processor.<handler_name>.bytesReceived
<instance>.<profile>.catalina.processor.<handler_name>.bytesSent
<instance>.<profile>.catalina.processor.<handler_name>.errorCount
<instance>.<profile>.catalina.processor.<handler_name>.maxTime
<instance>.<profile>.catalina.processor.<handler_name>.processingTime
<instance>.<profile>.catalina.processor.<handler_name>.requestCount

Example setups

The setups described in this chapter are provided as an example, without being part of the product or support. They can be found on the Kofax SignDoc Standard product downloads page.

Monitoring service sample setup
Collectd sample setup for collecting system metrics

This is an example how Collectd (via Docker containers) can be used to collect system metrics.
Glossary

**Cirrus**

The internal name of the SignDoc Standard part of the project. Manages signing packages, workflow and account / user administration.

**cluster**

The sum of instances working together as a whole to provide the application functionality. Consists of the application service instances, but also load balancer, cache service, log and monitoring service, etc.

**Collectd**

System statistics collection daemon [https://collectd.org/](https://collectd.org/)

**container**

Virtualization unit that encapsulates all necessary components to run an application on a system. In this document the term refers to Docker containers.

**Grafana**
Dashboard builder for visualizing monitoring data http://grafana.org/

**Graphite**

Set of monitoring components used to process monitoring information. In this document used to refer to the protocol defined by said components to transmit metrics http://graphite.readthedocs.org/en/latest/index.html

**InfluxDB**

Distributed time-series database used to store monitoring data https://influxdb.com

**metric**

A value to be monitored that is composed of name, value and timestamp.

**on premise deployment**

Form of deployment where the customer maintains control of all hardware and software components used to deploy the application. Only the application is provided, the customer administers the database server, servlet container, etc.

**private cloud deployment**

Form of cloud deployment where all resources of a cluster are allocated exclusively to one customer (tenant).

**public cloud deployment**

Form of cloud deployment where multiple customers (tenants) share the resources of one cluster.

**SignDoc Web**

The name of the SignDoc component that is used to process and sign a single document.

**tenant**

A customer or an organization that uses the services of an application and has multiple users. Multiple tenants can be set up on one system. The users of one tenant cannot access data of another tenant.
Important SignDoc Standard before version 2.1.0 was mainly configured with the configuration file cirrus.properties. This file moved to INSTALLDIR\_conf_templates\cirrus.properties with version 2.1.0.

Since SignDoc Standard 2.1.0, it is highly recommended to use the file INSTALLDIR\service_configuration.properties (instead of cirrus.properties) whenever it is required to configure SignDoc with a configuration file. Configurations set in this file are applied as Java System Property and have therefore highest precedence.

It is possible to create several accounts in SignDoc Standard for different tenants.

If you have several accounts you have to specify with which account you want to login for processing signing packages. This can be achieved by entering an accountid in the login panel.

As a tenant user it is very uncomfortable to enter an accountid each time for login.

A possibility is implemented to select in advance an account via tenant-specific URL.

Let’s assume that the SignDoc Standard application can be reached via https://www.signdoc.com:8083/cirrus.

We have two accounts in SignDoc Standard, one for customer "Customer One" and one for customer "Customer Two".

The administrator can configure in DNS (or for testing also in the local hosts file) that the SignDoc Standard server can be reached also via

- cone.signdoc.com for tenant "Customer One"

and

- ctwo.signdoc.com for the customer "Customer Two"

The domain name prefix cone and ctwo has to be configured in the SignDoc Standard application for identifying the appropriate account.

This can be done by the Server Administrator in the Account Details dialog using the field dnslabel.

For the tenant "Customer One" he has to enter cone in the entry field dnslabel and ctwo for tenant "Customer Two".

Important To activate this feature the administrator has to enter cirrus.tenant.url.supported=true in %CIRRUS_HOME%/conf/cirrus.properties. It is not activated by default! The DNS label must be unique for each tenant and at most 63 characters.

Implementation details
The SignDoc Standard application parses the domain name in the (login) URL and extracts the tenant-specific part to provide a request parameter dnslabel with this value.

**Example**

If the user calls URL cone.signdoc.com the application adds a request parameter dnslabel=cone to the call.

This dnslabel has to match with a dnslabel attribute in the ACCOUNT table for identifying an account.

In this case the SignDoc Standard application knows the requested account and the entry field for the accountid is omitted in the login dialog.