

DookuG Module Documentation

Table of Contents

Overview	2
1. Components	3
Backend	3
1. Services	3
1.1. DOOKUG-DOCUMENT Service	3
1.1.1. Document Generation	3
1.1.2. Document generation based on request body	5
1.1.3. Document generation based on request body, with metadata response	5
1.1.4. Document generation based on multipart form	5
1.1.5. Document generation based on multipart form, with metadata response	6
1.1.6. Document generation based on stored template	6
1.1.7. Document generation based on stored template, with metadata response	6
1.1.8. Query Document Metadata	7
1.1.9. Get Document	8
1.1.10. Electronic document signing (PDF)	8
2. OpenAPI Documentation	8
DookuG Module Client	8
1. Technology	8
2. Usage	9
3. Operation	9
3.1. Using Saxon-HE Engine in the Client	15
4. Error Handling	15
Configuration	15
1. Backend	16
1.1. DOOKUG-DOCUMENT service	16
1.1.1. dookug keys	16
1.2. Pdf Signature configuration	19
1.2.1. dookug keys	19
2. Client	25
Installation, Deployment	26
1. DOOKUG-DOCUMENT Service	26
1.1. Service configuration	26
1.2. Recommended K8S Configuration	28
1.3. Observability	29
1.3.1. Grafana dashboards	29
1.3.2. Health - startup/liveness/readiness	30

Additional informations	31
1. Template Cache	31
2. Helpers - Helper Functions for Use in Template Files	31
2.1. Built-in Helpers	31
2.2. Custom Helpers	31
2.2.1. Usage	32
2.2.2. Additional Helper Functions	37
3. Using Custom Fonts	37
4. Digitally Signing PDFs	39
4.1. Configuration	39
4.2. Changing Certificates	40
4.3. Visible signature (using the EU DSS ESIGN library)	40
4.4. Simple signature (using the PdfBox ESIGN library)	42
5. Keystore Cache for digital signature	42
Release notes	43
1. v0.4.0	43
2. v0.5.0	43
2.1. Changes / updates	43
2.2. Migration	44
3. v0.6.0	44
3.1. Changes / updates	44
3.2. Migration	44
4. v1.0.0	44
4.1. Changes / updates	44
4.2. Migration	45
5. v1.1.0	45
5.1. Changes / updates	45
5.2. Migration	46
6. v1.2.0	46
6.1. Changes / updates	46
6.2. Migration	46
7. v1.3.0	46
7.1. Changes / updates	46
7.2. Migration	46

Overview

The project aims to enable general template-based document generation for its clients.

1. Components

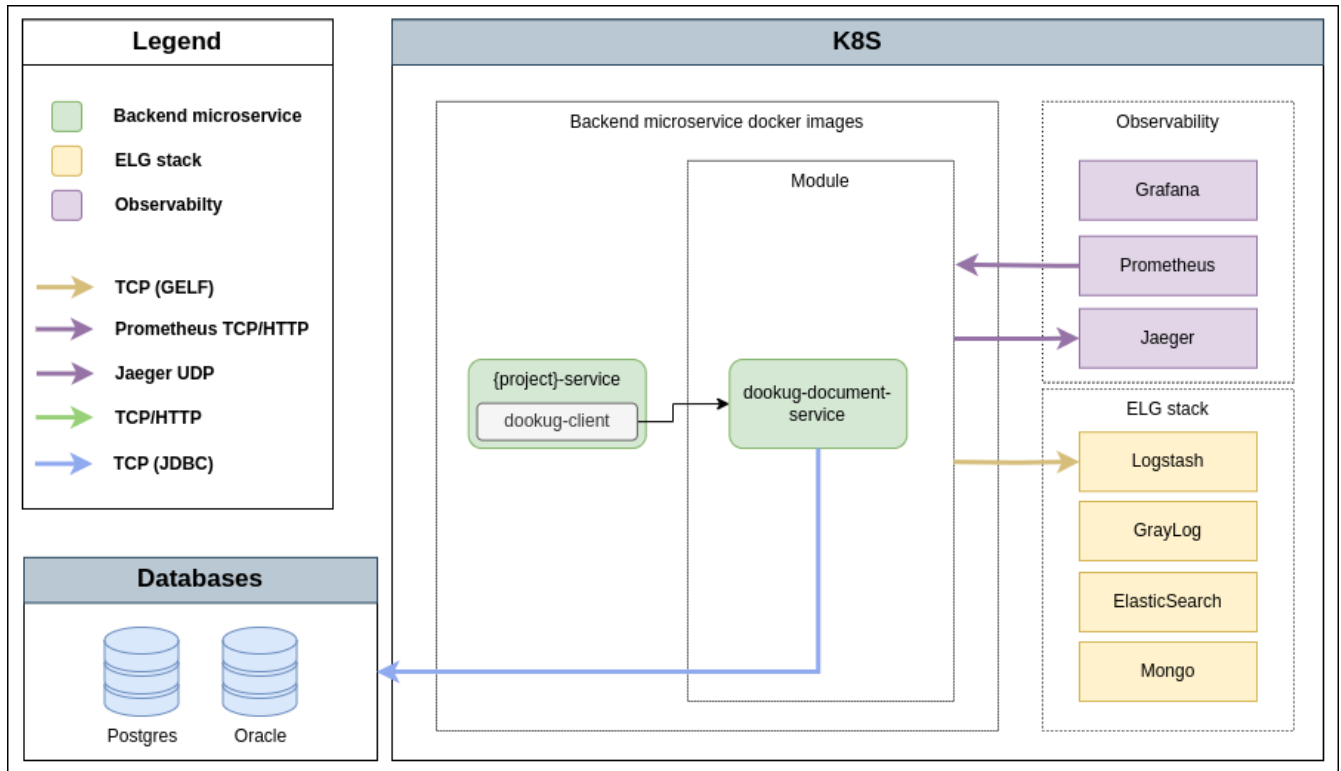


Figure 1. Architecture Diagram

Table 1. DookuG Backend 1.3.0-SNAPSHOT

Type	Module	Version
Internal	Oracle	21.3.0-xe
	PostgreSQL	dwh postgres image 0.10.0

Backend

1. Services

1.1. DOOKUG-DOCUMENT Service

1.1.1. Document Generation

The module provides multiple options for document generation. For each option, we can specify:

- The generator setup (`request.generatorSetup`):
 - Template engine for parameter substitution (NONE, HANDLEBARS) (`request.generatorSetup.templateEngine`)
 - Document generator engine (NONE, PDF_BOX, SAXON) (`request.generatorSetup.generatorEngine`)

- Response format (STRING, PDF) (`request.generatorSetup.responseFormat`)
- Storage type (NONE, DATABASE)
- Optional parameters (`request.generatorSetup.parameters` or `request.generatorSetup.parametersData.templateParameters` or `request.generatorSetup.parametersData.generatorParameters`)
- Optional digital signature profile (`request.generatorSetup.digitalSignatureProfile`)
- Optional template language, in case of inline SAXON generation (`request.generatorSetup.templateLanguage`)
- Template, in case of a stored one, specifying the following attributes for finding it in cache or the database, or for SAXON generation, choosing the language within the XSLT:
 - `templateName`
 - `templateLanguage`
 - Optional `validityDate`
- The templates (`request.templates`), as list of `TemplateType` objects, in case of inline generation, specifying the `templateName`, `templateContent` in base64 binary format and the `initail` flag, representing the initial template and other template parts

Firstly, the generator setup is validated based on the response format and the generator engine:

- If the response format is PDF, then the generator engine can only be PDF_BOX or SAXON
- If the response format is STRING, then the generator engine can only be NONE

Secondly, in the template compilation phase the following scenarios can occur, based on the template engine, the parameters or parametersData:

- If the template engine is HANDLEBARS, then compilation takes place:
 - If parameters are provided (key - value pairs) OR
 - If parametersData is provided (json key - value pairs in base64 binary format, can be zipped as well (GZIP)) OR
 - No parameters are provided (compilation with empty context)
- If the template engine is NONE:
 - You can send only one template in the templates list with no parameters or parametersData
 - No compilation will occur

Thirdly, on document generation, optional signing and optional storage:

- The document stream is created if the generator engine is PDF_BOX or SAXON:
 - PDF_BOX:
 - The document is created by the compiled html
 - Optionally the document can be signed if the digital signature profile was provided in the request

- SAXON:

- With this generator the compilation happens during generation by the provided parametersData (`request.generatorSetup.parametersData.generatorParameters`)
- Optionally the document can be signed if the digital signature profile was provided in the request
- NOTE: It is possible to combine SAXON with HANDLEBARS. This way you can compile XSLT templates dynamically:
 - Set the template engine to HANDLEBARS and the generator to SAXON
 - Provide XSLT template parts in the templates list (main template can contain HANDLEBARS placeholders)
 - Provide XSLT template parameters in the generatorSetup.parametersData

- Storage, based on storage method type

- DATABASE:

- The document is saved with parameters, parametersData or null parametersData and the document is returned

- NONE:

- The document is returned

1.1.2. Document generation based on request body

```
POST /internal/dookug/document/generate/inline
```

During generation, we submit the initial template and the data associated with the generation in the request body: template and engine processing the document, response format, and document storage method.

In the response, we receive the generated document and the filename in the HTTP header.

1.1.3. Document generation based on request body, with metadata response

```
POST /internal/dookug/document/generate/inline/metadata
```

There's an option to receive metadata describing the document instead of the document itself. In this case, the request URI and HTTP header differ when submitting the request.

1.1.4. Document generation based on multipart form

```
POST /internal/dookug/document/generate/inline/multipart
```

To generate, we need to submit the initial template and the data associated with the generation. The latter matches the data sent for document generation based on the request body.

In the response, we receive the generated document and the filename in the HTTP header.

1.1.5. Document generation based on multipart form, with metadata response

```
POST /internal/dookug/document/generate/inline/multipart/metadata
```

There's also an option to receive metadata describing the document instead of the document itself. In this case, the request URI and HTTP header differ when submitting the request.

1.1.6. Document generation based on stored template

```
POST /internal/dookug/document/generate/storedTemplate
```

To generate, we need to submit the name of the template from which the document is generated, along with the data associated with the generation. The latter matches the data sent for document generation based on the request body, supplemented with how the template is stored.

In the response, we receive the generated document and the filename in the HTTP header.

1.1.7. Document generation based on stored template, with metadata response

```
POST /internal/dookug/document/generate/storedTemplate/metadata
```

There's an option to receive metadata describing the document instead of the document itself. In this case, the request URI and HTTP header differ when submitting the request.

The template key stored in the database consists of the `templateName` and `templateLanguage` values.

Saving the document depends on the `documentStorageMethod` parameter. It can take two values: `NONE` and `DATABASE`. If `NONE` is specified, the document is not saved, and therefore cannot be queried later. In case of `DATABASE`, the generated document is saved in a database table, from where it can be retrieved later. Other data related to the document are also saved in the database:

- identifier of the initial template - if the template is not saved, this parameter is not filled
- filename of the generated file - generated from the unique identifier of the document, the name of the initial template, and the timestamp of the generation long value
- file format
- document status - DONE, FAILED, PENDING, SYNCING
- parameters related to the document

- document storage format - in case of the field DATABASE, this field is also filled with DATABASE value

During generation, the initial template is provided in any form of the response, the generated file is received, or metadata describing the document is received, as a DocumentMetadataResponse type object.

1.1.8. Query Document Metadata

```
POST /internal/dookug/document/storedTemplate/metadata/query
```

The purpose of querying document metadata is to retrieve document information that meets the specified filtering criteria.

The endpoint supports pagination, meaning data can be retrieved across multiple pages. In the request, you can specify which page of data and how many elements per page you want to retrieve. Accordingly, the response includes total count of elements and the number of pages they span. If not specified, the endpoint defaults to returning the first 15 elements.

The following filtering criteria can be used:

- templateId - identifier of the template used for document generation
- status - status of the document
- format - file format of the document
- storageMethod - storage method of the document
- storageId - unique identifier of the document storage
- filename - name of the document file

Sorting parameters can be:

- filename
- documentStorageMethod
- format
- status

For sorting, you can specify whether to sort in ascending or descending order for each parameter individually. In addition to the mentioned sorting options, there is a default sorting by document identifier.

If documents are found based on the submitted parameters, the response returns a list of up to 100 elements.

1.1.9. Get Document

```
GET /internal/dookug/document/content/{documentId}
```

The purpose of this endpoint is to retrieve a previously generated and saved document based on the provided identifier.

If no document is found for the submitted identifier, an ENTITY_NOT_FOUND error is returned.

In the response - for an existing document identifier - the generated document is returned, and the file name is included in the HTTP headers.

1.1.10. Electronic document signing (PDF)

```
POST /internal/dookug/document/sign/inline/multipart
```

The document received in the request is digitally signed and returned in the response. The request must include the file to be signed and the name of the signing profile. The signing process is performed synchronously based on the parameters configured in the module for the given profile. The file is not stored on the server at any point.

In the response, we receive the signed document and the filename in the HTTP header. The filename can be set in the request.

2. OpenAPI Documentation

Download OpenAPI Files:

- dookug-rest-service
 - [yaml](#)
 - [json](#)

DookuG Module Client

The client aims to support management of the DookuG module, which encompasses various server-side methods.

1. Technology

- Java 17
- JEE 10 (There is a client for supporting JEE 8)
- Eclipse Microprofile 4.1 and 5.0

2. Usage

Using the client requires only an entry in the pom.xml file, depending on the desired JEE version:

```
<dependency>
  <groupId>hu.icellmobilsoft.dookug.client</groupId>
  <artifactId>dookug-client-jee10</artifactId>
  <version>${version.dookug.client}</version>
</dependency>
```

or

```
<dependency>
  <groupId>hu.icellmobilsoft.dookug.client</groupId>
  <artifactId>dookug-client-jee8</artifactId>
  <version>${version.dookug.client}</version>
</dependency>
```

followed by an @Inject,

```
@Inject
private DookugClient dookugClient;
```

which defines client calls to all endpoints and allows the use of its offered methods. The API calls are made using the [MicroProfile Rest Client](#), so configurations can be set using the familiar microprofile-config. (see below)



It is the responsibility of the client application to mount resources under the `/home/icellmobilsoft/resources` mount point within the container that it uses, which are referenced in the templates!

3. Operation

The client currently has multiple methods to generate documents, retrieve metadata of the created documents, and fetch the generated document itself.

During document generation, some methods return the generated document:

```
public GeneratedDocumentDto postDocumentGenerateEntityBody(Collection<TemplateType>
templates, Collection<ParameterType> parameters) throws BaseException;

public GeneratedDocumentDto postDocumentGenerateEntityBody(Collection<TemplateType>
templates, ParametersDataType parametersData) throws BaseException;
```

```

public GeneratedDocumentDto postDocumentGenerateEntityBody(Collection<TemplateType>
templates) throws BaseException;

public GeneratedDocumentDto postDocumentGenerateMultipart(InputStream template,
Collection<ParameterType> parameters) throws BaseException;

public GeneratedDocumentDto postDocumentGenerateMultipart(InputStream template,
ParametersDataType parametersData) throws BaseException;

public GeneratedDocumentDto postDocumentGenerateMultipart(InputStream template) throws
BaseException;

public GeneratedDocumentDto postDatabaseStoredTemplateDocumentGenerate(String
templateName, OffsetDateTime templateValidity, Collection<ParameterType> parameters)
throws BaseException;

public GeneratedDocumentDto postDatabaseStoredTemplateDocumentGenerate(String
templateName, OffsetDateTime templateValidity, ParametersDataType parametersData)
throws BaseException;

```

It is important to note that `ParametersDataType` type parameters can be created manually, but a `ParametersDataBuilder` helper class is provided to configure the desired settings using a fluent API. This was required by the SAXON template engine, as additional parameters are necessary for generating XSLT templates, without which documents cannot be generated. The desired configuration can be easily extracted from the builder for the SAXON engine by calling the `getSaxonParameters(byte[], byte[], boolean)` method, where the FOP configuration content, the XML dataset, and the compression status of the XML can be specified, or through overloaded methods, just the content of the XML file.

```

GeneratedDocumentDto resp = client.postDatabaseStoredTemplateDocumentGenerate(
    TemplateLanguageType.HU, ①
    ParametersDataBuilder.getSaxonParameters(
        FileUtil.readFileFromResource(DocumentServiceTestConstant
.XSLT_TEMPLATE_PARAMS).getBytes(StandardCharsets.UTF_8)); ②

```

① The language of the template

② The content of the XML dataset (uncompressed in this case)

These methods return a `GeneratedDocumentDTO` upon a successful call, which contains the generated filename, the generated object as a stream, and the HTTP status code.

Parameters must include the list of templates and the parameters if the templates contain variables that need to be replaced with values, such as in the case of the HANDLEBARS template engine. The parameters can be key-value pairs or a JSON object that essentially contains these key-value pairs.

The methods differ in the way the template for document generation is provided. In the `postDocumentGenerateEntityBody()` methods, the structure used as a template is sent in the request body. In the `postDocumentGenerateMultipart()` methods, the template is provided as a form

parameter, `InputStream`. In the `postDatabaseStoredTemplateDocumentGenerate()` methods, only the template name needs to be provided, as the endpoint handles it according to the specified storage method.

Additionally, methods are available for generating documents where only the metadata of the created document is returned instead of the document itself:

```
public DocumentMetadataResponse postDocumentGenerateEntityBodyMetadata(Collection<TemplateType> templates, Collection<ParameterType> parameters) throws BaseException;

public DocumentMetadataResponse postDocumentGenerateEntityBodyMetadata(Collection<TemplateType> templates, ParametersDataType parametersData) throws BaseException;

public DocumentMetadataResponse postDocumentGenerateEntityBodyMetadata(Collection<TemplateType> templates) throws BaseException;

public DocumentMetadataResponse postDocumentGenerateMultipartMetadata(InputStream template, Collection<ParameterType> parameters) throws BaseException;

public DocumentMetadataResponse postDocumentGenerateMultipartMetadata(InputStream template, ParametersDataType parametersData) throws BaseException;

public DocumentMetadataResponse postDocumentGenerateMultipartMetadata(InputStream template) throws BaseException;

public DocumentMetadataResponse postDatabaseStoredTemplateDocumentGenerateMetadata(String templateName, OffsetDateTime templateValidity, Collection<ParameterType> parameters) throws BaseException;

public DocumentMetadataResponse postDatabaseStoredTemplateDocumentGenerateMetadata(String templateName, OffsetDateTime templateValidity, ParametersDataType parametersData) throws BaseException;

public DocumentMetadataResponse postStoredTemplateDocumentGenerateMetadata(String templateName, OffsetDateTime templateValidity, TemplateStorageMethodType templateStorageMethodType, Collection<ParameterType> parameters, ParametersDataType parametersData) throws BaseException;
```

These methods return a `DocumentMetadataResponse` upon a successful call, which includes the unique identifier of the document, filename, storage method, format, and status.

Similarly, the client calls are distinguished in the same way by how the template is provided.

To view already generated files, there is an option to query the metadata of the documents:

```
public DocumentMetadataQueryResponse postDocumentMetadataQuery(DocumentMetadataQueryParamsType queryParams) throws BaseException;

public DocumentMetadataQueryResponse postDocumentMetadataQuery
```

```
(DocumentMetadataQueryRequest queryRequest) throws BaseException;
```

The query returns a `DocumentMetadataQueryResponse` upon a successful call, which contains the metadata of documents matching the specified parameters, as well as pagination data: total number of items, total number of pages, page number, and the number of items returned in the query.

Filtering conditions can include the document name, format, storage method, template identifier from which the document was generated, the identifier of the document's storage, and the status of the document.

It is also possible to query a previously generated document:

```
public GeneratedDocumentDto getDocumentContent(String documentId) throws  
BaseException;
```

The response returns the generated document as a stream, the filename, and the HTTP status code upon a successful call.

The unique identifier of the document must be provided as a parameter.

The client also offers additional configuration options that can influence the generation process:

```
setGeneratorEngineType()
```

```
setTemplateEngineType()
```

```
setResponseFormat()
```

```
setDocumentStorageMethodType()
```

```
setDigitalSigningType()
```

The `setGeneratorEngineType()` allows setting the engine used for output generation, which currently can be:

- `PDF_BOX` — uses [Apache PdfBox](<https://pdfbox.apache.org/>)
- `SAXON` — uses [Saxon HE](<https://github.com/Saxonica/Saxon-HE/>)
- `NONE`

The `setTemplateEngineType()` allows setting the template 'type', which currently can be:

- **HANDLEBARS** — uses [Handlebars](https://handlebarsjs.com/)
- **NONE**

The **setResponseFormat()** allows setting the response format:

- **PDF**
- **STRING**

The **setDocumentStorageMethodType()** allows setting the document storage method:

- **NONE**
- **DATABASE**

The **setDigitalSigningType(digitalSigningType)** allows controlling whether the generated PDF document should have a digital signature. The **digitalSigning** expects three additional parameters:

- **signatureName** - the name of the signature (optional)
- **signatureReason** - the reason for signing (optional)
- **keyAlias** - the identifier of the key in the keystore, used to identify the signing key (optional, but recommended as it defaults to searching for the **test** key)

If no settings are specified, the default values are **PDF_BOX** + **HANDLEBARS** + **PDF** + **NONE** without a digital signature.

The **postDocumentGenerateEntityBody()** methods call the following REST endpoint in the module:

```
POST /internal/dookug/document/generate/inline
```

The client sends:

- **ContextType**
- the received **TemplateType** list
- the received **ParameterType** list
- the **GeneratorSetup** object, which can be controlled by the client's **set** methods.

If the request is correct, the generated object is returned.



For Multipart and StoredTemplate clients, the process is the same, differing only in the REST API calls.

The **postDocumentMetadataQuery()** methods call the following REST endpoint in the module:

```
POST /internal/dookug/document/storedTemplate/metadata/query
```

The client sends:

- `ContextType`
- the received filtering conditions
- pagination parameters
- sorting settings

If the request is correct, the metadata of the documents matching the specified parameters is returned.

Example of client usage:

```
@Inject
private DookugClient dookugClient;
...
//template object
TemplateType template = new TemplateType().withTemplateName("main")
).withTemplateContent("DookuG client simple test with parameters first: [{{first}}],
second: [{{second}}]").getBytes(StandardCharsets.UTF_8));

//parameters
ParameterType parameter1 = new ParameterType().withKey("first").withValue("első");
ParameterType parameter2 = new ParameterType().withKey("second").withValue
("í189öüóóúúáé-.,<>#&@{};*~ß$");
...
client.setResponseFormatType(ResponseFormatType.STRING);
client.setGeneratorEngineType(GeneratorEngineType.NONE);
GeneratedDocumentDto response = dookugClient.postDocumentGenerateEntityBody(List.of
(template), List.of(parameter1,parameter2));
```

Or similarly, generating a document but with a PDF document format, multipart input, and returning the metadata:

```
@Inject
private DookugClient dookugMultipartClient;
...
//template as byte array
byte[] template = "DookuG client simple test with parameters first: [{{first}}],
second: [{{second}}]").getBytes(StandardCharsets.UTF_8);

//parameters
ParameterType parameter1 = new ParameterType().withKey("first").withValue("első");
ParameterType parameter2 = new ParameterType().withKey("second").withValue
("í189öüóóúúáé-.,<>#&@{};*~ß$");
...
client.setResponseFormatType(ResponseFormatType.PDF); //ez a default
client.setGeneratorEngineType(GeneratorEngineType.PDF_BOX); //ez a default
client.setTemplateEngineType(GeneratorEngineType.HANDLEBARS); //ez a default
DocumentMetadataResponse response = dookugMultipartClient
.postDocumentGenerateMultipartMetadata(new ByteArrayInputStream(template), List.of
```

```
(parameter1, parameter2));
```

3.1. Using Saxon-HE Engine in the Client

To use the Saxon-HE engine in the client, an XSLT template is required for generating a PDF file from an XML. In this case, only PDF can be the output format. You also need to provide a `fop-config.xml` file in the request, which helps regulate the use of fonts, for example.

```
<?xml version="1.0" encoding="UTF-8"?>
<fop version="1.0">
  <renderers>
    <renderer mime="application/pdf">
      <fonts>
        <!-- TTF fonts -->
        <font kerning="yes" embed-
url="/home/icellmobilsoft/fonts/Roboto/Roboto-Regular.ttf">
          <font-triplet name="Roboto" style="normal" weight="normal" />
        </font>
        <font kerning="yes" embed-
url="/home/icellmobilsoft/fonts/Roboto/Roboto-Bold.ttf">
          <font-triplet name="Roboto" style="normal" weight="bold" />
        </font>
      </fonts>
    </renderer>
  </renderers>
</fop>
```

Handlebars can also be used with SAXON, where you can substitute the usual `{{VARIABLE}}` variables with desired text parts, as well as create nested templates (this is mainly used for that).

The major change from other engines is that you need to specify the `XML` as a data source in the `generatorSetup` in the SAXON case, in addition to other fields:

- XML: as data source
- XSLT: as template
- fopConfig: transformer configuration

4. Error Handling

The client can only return a `BaseException`, but if a `RestClientResponseException` is received during the API call, it will return the wrapped `BaseException` contained within it!

Configuration

1. Backend

The module's configuration is managed via MicroProfile Config, which allows specifying necessary values in multiple ways.

MicroProfile Config can retrieve a given key from all available sources and uses the highest priority value.

The base configuration is provided via `project-defaults.yml`, which can be extended and may vary per environment. It's not necessary to specify every value; only those that differ from the default settings.

Possible modes in order of priority:

- System variables
- Environment variables

1.1. DOOKUG-DOCUMENT service

The meanings of the emojis used in the table:

🚀 - meaning that it is a startup parameter.

🔄 - meaning that this parameter can be overridden during runtime

1.1.1. dookug keys

Key	Source	Description	Default value	Since	Features
dookug.client.document	hu.icellmobilsoft.dookug.api.dto.Constants.ConfigKeys.Client	Client configuration In the application using the DookuG client, you need to set: - the server REST URL (e.g., dookug.client.document/mp-rest/url: http://localhost:8082) - optionally the REST connectTimeout (e.g., dookug.client.document/mp-rest/connectTimeout: 5000) - optionally the REST readTimeout (e.g., dookug.client.document/mp-rest/readTimeout: 60000)	-	0.1.0	

Key	Source	Description	Default value	Since	Features
dookug.service.cache.keystore.enabled	hu.icellmobilsoft.dookug.api.dto.constants.ConfigKeys.Cache.Key store	Does the module use keystore caching for document signing?	true	1.1.0	
dookug.service.cache.keystore.enablestatis tic	hu.icellmobilsoft.dookug.api.dto.constants.ConfigKeys.Cache.Key store	Metrics related to the Template cache should be generated. By default, they are not generated.	false	1.1.0	
dookug.service.cache.keystore.ttl	hu.icellmobilsoft.dookug.api.dto.constants.ConfigKeys.Cache.Key store	How long until the system invalidates the cache content. By default, 1440 minutes.	1440	1.1.0	
dookug.service.cache.template.enabled	hu.icellmobilsoft.dookug.api.dto.constants.ConfigKeys.Cache.Template	Does the module use template caching?	false	0.6.0	
dookug.service.cache.template.enablestatis tic	hu.icellmobilsoft.dookug.api.dto.constants.ConfigKeys.Cache.Template	Metrics related to the Template cache should be generated. By default, they are not generated.	false	0.5.0	
dookug.service.cache.template.ttl	hu.icellmobilsoft.dookug.api.dto.constants.ConfigKeys.Cache.Template	How long until the system invalidates the cache content. By default, 60 minutes.	60	0.5.0	

Key	Source	Description	Default value	Since	Features
dookug.service.cache.xslt.enabled	hu.icellmobilsoft.dookug.api.dto.constants.ConfigKeys.Cache.Xslt	Does the module use template caching for saxon?	true	1.3.0	
dookug.service.cache.xslt.enablestatistic	hu.icellmobilsoft.dookug.api.dto.constants.ConfigKeys.Cache.Xslt	Metrics related to the Template cache should be generated. By default, they are not generated.	false	1.3.0	
dookug.service.cache.xslt.ttl	hu.icellmobilsoft.dookug.api.dto.constants.ConfigKeys.Cache.Xslt	How long until the system invalidates the cache content. By default, 1440 minutes.	1440	1.3.0	
dookug.service.engine.handlebars.escapingstrategy	hu.icellmobilsoft.dookug.api.dto.constants.ConfigKeys.Handlebars.EscapingStrategy	Handlebars configuration Configuration storing the <code>com.github.jknack.handlebars.EscapingStrategy</code> key.	In the Handlebars engine, HTML will be the default strategy if no value is configured with this key.	0.1.0	
dookug.service.engine.handlebars.helper.javascript.directory	hu.icellmobilsoft.dookug.api.dto.constants.ConfigKeys.Handlebars.Helper	Handlebars configuration Stores the path to the directory containing JavaScript files with Handlebars handlers in the Docker container.	/home/icellmobilsoft/handlebars/helper/js	0.1.0	
dookug.service.engine.saxon.fopconfig	hu.icellmobilsoft.dookug.api.dto.constants.ConfigKeys.Saxon	Saxon configuration Path to the Fop config file within the container	/home/icellmobilsoft/fop-config/fop-config.xml	0.1.0	

Key	Source	Description	Default value	Since	Features
dookug.service.engine.saxon.xslt.language.default	hu.icellmobilsoft.dookug.api.dto.constants.ConfigKeys.Saxon	Saxon configuration Default language	HU	0.1.0	
dookug.service.engine.saxon.xslt.language.variable	hu.icellmobilsoft.dookug.api.dto.constants.ConfigKeys.Saxon	Saxon configuration The name of the language variable in the XSLT template	lang	0.1.0	
dookug.service.interface.parametersdata.gzipped	hu.icellmobilsoft.dookug.api.dto.constants.ConfigKeys.Interface	Interface configuration Logical config key. The module expects the incoming "parametersData" field in the request to be compressed using gzip.	false	0.1.0	

1.2. Pdf Signature configuration

The meanings of the emojis used in the table:

🚧 - meaning that it is a startup parameter.

🔄 - meaning that this parameter can be overridden during runtime

1.2.1. dookug keys

Key	Source	Description	Default value	Since	Features
dookug.service.engine.pdf.digital.sign.{0}.	hu.icellmobilsoft.dookug.engine.pdfbox.config.ConfigKeys.PdfSignature	the (root) key to retrieve the signature profile, where the first parameter is the profile name: {@value #DOOKUG_SERVICE_ENGINE_PDF_DIGITALSIGN_PROFILE_0}			

Key	Source	Description	Default value	Since	Features
dookug.service.engine.pdf.digital.sign.{0}.dss.certificatePermission	hu.icellmobilsoft.dookug.engine.pdfbox.config.ConfigKeys.PdfSignature.DSS	The certificate permission (Pdf will be either 'certificated' or - without this - 'signed') {@value #CERTIFICATE_PERMISSION}			
dookug.service.engine.pdf.digital.sign.{0}.dss.digestAlgorithm	hu.icellmobilsoft.dookug.engine.pdfbox.config.ConfigKeys.PdfSignature.DSS	The digest algorithm {@value #DIGEST_ALGORITHM}			
dookug.service.engine.pdf.digital.sign.{0}.dss.encryptionAlgorithm	hu.icellmobilsoft.dookug.engine.pdfbox.config.ConfigKeys.PdfSignature.DSS	The encryption algorithm {@value #ENCRYPTION_ALGORITHM}			
dookug.service.engine.pdf.digital.sign.{0}.dss.hintText	hu.icellmobilsoft.dookug.engine.pdfbox.config.ConfigKeys.PdfSignature.DSS	Text to be displayed in hint field {@value #HINT_TEXT}			
dookug.service.engine.pdf.digital.sign.{0}.dss.imageFile	hu.icellmobilsoft.dookug.engine.pdfbox.config.ConfigKeys.PdfSignature.DSS	Image to be placed in signature block {@value #IMAGE_FILE}			

Key	Source	Description	Default value	Since	Features
dookug.service.engine.pdf.digital.sign.{0}.dss.labelHint	hu.icellmobilsoft.dookug.engine.pdfbox.config.ConfigKeys.PdfSignature.DSS	Label for the 'hint' row {@value #LABEL_HINT}			
dookug.service.engine.pdf.digital.sign.{0}.dss.labelSignee	hu.icellmobilsoft.dookug.engine.pdfbox.config.ConfigKeys.PdfSignature.DSS	Label for the 'signee' row {@value #LABEL_SIGNEE}			
dookug.service.engine.pdf.digital.sign.{0}.dss.labelTimestamp	hu.icellmobilsoft.dookug.engine.pdfbox.config.ConfigKeys.PdfSignature.DSS	Label for the 'timestamp' row {@value #LABEL_TIMESTAMP}			
dookug.service.engine.pdf.digital.sign.{0}.dss.position.left	hu.icellmobilsoft.dookug.engine.pdfbox.config.ConfigKeys.PdfSignature.DSS	X coordinate of the signature block in cm {@value #POSITION_LEFT}			
dookug.service.engine.pdf.digital.sign.{0}.dss.position.top	hu.icellmobilsoft.dookug.engine.pdfbox.config.ConfigKeys.PdfSignature.DSS	Y coordinate of the signature block in cm {@value #POSITION_TOP}			

Key	Source	Description	Default value	Since	Features
dookug.vice.engine.pdf.digital.sign.{0}.dss.showOnPage	hu.icellmobilsoft.dookug.engine.pdfbox.config.ConfigKeys.PdfSignature.DSS	Page where the signature block should be placed. [-1] for last page, 0: invisible {@value #SHOW_ON_PAGE}			
dookug.vice.engine.pdf.digital.sign.{0}.dss.showSignee	hu.icellmobilsoft.dookug.engine.pdfbox.config.ConfigKeys.PdfSignature.DSS	Show 'signee' row {@value #SHOW_SIGNEE}			
dookug.vice.engine.pdf.digital.sign.{0}.dss.skipTimestampOnError	hu.icellmobilsoft.dookug.engine.pdfbox.config.ConfigKeys.PdfSignature.DSS	Skip timestamp on error (only raise log entry instead of exception) {@value #SKIP_TIMESTAMP_ON_ERROR}			
dookug.vice.engine.pdf.digital.sign.{0}.dss.timezone	hu.icellmobilsoft.dookug.engine.pdfbox.config.ConfigKeys.PdfSignature.DSS	Use specific timezone for time info, e.g. Europe/Budapest {@value #TIMEZONE}			
dookug.vice.engine.pdf.digital.sign.{0}.dss.trustedCertificates	hu.icellmobilsoft.dookug.engine.pdfbox.config.ConfigKeys.PdfSignature.DSS	URL list for getting trusted certificates {@value #TRUSTED_CERTIFICATES}			

Key	Source	Description	Default value	Since	Features
dookug.service.pdf.digital.sign.{0}.dss.tspSources	hu.icellmobilsoft.dookug.engine.pdfbox.config.ConfigKeys.PdfSignature.DSS	Use specific time stamping authority as source (if multiple given, will be used in given order as fallback) {@value #TSP_SOURCES}			
dookug.service.pdf.digital.sign.{0}.dss.useHint	hu.icellmobilsoft.dookug.engine.pdfbox.config.ConfigKeys.PdfSignature.DSS	Add hint row in signature table {@value #USE_HINT}			
dookug.service.pdf.digital.sign.{0}.dss.useLT	hu.icellmobilsoft.dookug.engine.pdfbox.config.ConfigKeys.PdfSignature.DSS	Use PAdES profile with long-term validation material {@value #USE_LT}			
dookug.service.pdf.digital.sign.{0}.dss.useLTA	hu.icellmobilsoft.dookug.engine.pdfbox.config.ConfigKeys.PdfSignature.DSS	Use PAdES profile with long term availability and integrity of validation material {@value #USE_LTA}			
dookug.service.pdf.digital.sign.{0}.dss.useTimestamp	hu.icellmobilsoft.dookug.engine.pdfbox.config.ConfigKeys.PdfSignature.DSS	Include signed timestamp in the signature table {@value #USE_TIMESTAMP}			

Key	Source	Description	Default value	Since	Features
dookug.service.engine.pdf.digital.sign.{0}.dss.width	hu.icellmobilsoft.dookug.engine.pdfbox.config.ConfigKeys.PdfSignature.DSS	Width of the signature block in cm {@value #WIDTH}			
dookug.service.engine.pdf.digital.sign.{0}.keyAlias	hu.icellmobilsoft.dookug.engine.pdfbox.config.ConfigKeys.PdfSignature	key alias configuration key of signature profile {@value #KEY_ALIAS}			
dookug.service.engine.pdf.digital.sign.{0}.keystore	hu.icellmobilsoft.dookug.engine.pdfbox.config.ConfigKeys.PdfSignature	keystore configuration key of signature profile {@value #KEYSTORE}			
dookug.service.engine.pdf.digital.sign.{0}.keystorePass	hu.icellmobilsoft.dookug.engine.pdfbox.config.ConfigKeys.PdfSignature	keystore password configuration key of signature profile {@value #KEYSTORE_PASSWORD}			
dookug.service.engine.pdf.digital.sign.{0}.keystoreType	hu.icellmobilsoft.dookug.engine.pdfbox.config.ConfigKeys.PdfSignature	keystore type configuration key of signature profile {@value #KEYSTORE_TYPE}			
dookug.service.engine.pdf.digital.sign.{0}.name	hu.icellmobilsoft.dookug.engine.pdfbox.config.ConfigKeys.PdfSignature	default signature name configuration key of signature profile {@value #NAME}			

Key	Source	Description	Default value	Since	Features
dookug.service.engine.pdf.digital.sign.{0}.pdfbox.signatureAlgorithm	hu.icellmobilsoft.dookug.engine.pdfbox.config.ConfigKeys.PdfSignature.PdfBox	The signature algorithm {@value #SIGNATURE_ALGORITHM}			
dookug.service.engine.pdf.digital.sign.{0}.reason	hu.icellmobilsoft.dookug.engine.pdfbox.config.ConfigKeys.PdfSignature	default signature reason configuration key of signature profile {@value #REASON}			
dookug.service.engine.pdf.digital.sign.{0}.useEuDssSig	hu.icellmobilsoft.dookug.engine.pdfbox.config.ConfigKeys.PdfSignature	use the eu-dss-sig library for pdf signature {@value #USE_EUDSSSIG}			

2. Client

The operation of the client can be influenced using MicroProfile Config.

```
dookug:
  client:
    document/mp-rest/url: http://localhost:8082
    document/mp-rest/connectTimeout: 5000 #millisec
    document/mp-rest/readTimeout: 60000 #millisec
```

It is advisable to specify the following parameters for the client operation:

Parameter Key	ENV	Mandatory	Description
dookug.client.document/mp-rest/url	DOOKUG_CLIENT_DOCUMENT_MP_REST_URL	yes	Base URL of the DookuG module
dookug.client.document/mp-rest/connectTimeout	DOOKUG_CLIENT_DOCUMENT_MP_REST_CONNECT_TIMEOUT		Connection Timeout (defaults to 5 seconds if unspecified)

Parameter Key	ENV	Mandatory	Description
dookug.client.document/mp-rest/readTimeout	DOOKUG_CLIENT_DOCUMENT_MP_REST_READ_TIMEOUT		Read Timeout (defaults to 1 minute if unspecified)
dookug.service.cache.template.ttl	DOOKUG_SERVICE_CACHE_TEMPLATE_TTL	no	Template cache expiration time in minutes, default is 60 minutes
dookug.service.cache.template.enablestatistic	DOOKUG_SERVICE_CACHE_TEMPLATE_ENABLE_STATISTIC	no	Whether to generate metrics for caching, defaults to false

Additional parameters can also be specified, as outlined in the [Microprofile RestClient documentation](<https://download.eclipse.org/microprofile/microprofile-rest-client-2.0/microprofile-rest-client-spec-2.0.html#mpconfig:~:text=Client%20CDI%20Support-,Support%20for%20MicroProfile%20Config,-Configuration%20Keys>).

Installation, Deployment

The Dookug module must be accessible in the environments of the project(s) that intend to utilize the service. Each instance of the service, along with its infrastructure requirements, should be installed and configured for every environment (development/test/production).

This includes:

- a relational database (Oracle or Postgres)

1. DOOKUG-DOCUMENT Service

1.1. Service configuration

Oracle

```
ORACLE_DS_CONNECTION_URL=jdbc:oracle:thin:@module-dookug-oracle:1521/xepdb1
ORACLE_DS_USERNAME=dookug
ORACLE_DS_PASSWORD=developer
ORACLE_DS_MIN-POOL-SIZE=1
ORACLE_DS_MAX-POOL-SIZE=20
```

Postgres

```
POSTGRESQL_DS_CONNECTION_URL=jdbc:postgresql://module-dookug-postgresdb:5432/dookug_db?currentSchema=dookug
POSTGRESQL_DS_USERNAME=postgres
POSTGRESQL_DS_PASSWORD=postgres
```

```
POSTGRESQL_DS_MIN-POOL-SIZE=1
POSTGRESQL_DS_MAX-POOL-SIZE=20
```

Interface

```
DOOKUG_SERVICE_INTERFACE_PARAMETERSDATA_GZIPPED=true
```

PDF signing

```
DOOKUG_SERVICE_ENGINE_PDF_DIGITALSIGN_DEFAULT_NAME="Original document"
DOOKUG_SERVICE_ENGINE_PDF_DIGITALSIGN_DEFAULT_REASON="Certified by Dookug"
DOOKUG_SERVICE_ENGINE_PDF_DIGITALSIGN_DEFAULT_KEYSTORE=/home/icellmobilsoft/keys/keystore.p12
DOOKUG_SERVICE_ENGINE_PDF_DIGITALSIGN_DEFAULT_KEYALIAS='test'
DOOKUG_SERVICE_ENGINE_PDF_DIGITALSIGN_DEFAULT_KEYSTOREPASS=123456
DOOKUG_SERVICE_ENGINE_PDF_DIGITALSIGN_DEFAULT_KEYSTORETYPE=PKCS12
DOOKUG_SERVICE_ENGINE_PDF_DIGITALSIGN_DEFAULT_USEDSSSIG=true ①
DOOKUG_SERVICE_ENGINE_PDF_DIGITALSIGN_DEFAULT_DSS_IMAGEFILE=/home/icellmobilsoft/pdfs/sign/sample/signature.png ②
DOOKUG_SERVICE_ENGINE_PDF_DIGITALSIGN_DEFAULT_DSS_SHOWONPAGE=1 ③
DOOKUG_SERVICE_ENGINE_PDF_DIGITALSIGN_DEFAULT_DSS_POSITION_LEFT=18
DOOKUG_SERVICE_ENGINE_PDF_DIGITALSIGN_DEFAULT_DSS_POSITION_WIDTH=2
DOOKUG_SERVICE_ENGINE_PDF_DIGITALSIGN_DEFAULT_DSS_USETIMESTAMP=false
DOOKUG_SERVICE_ENGINE_PDF_DIGITALSIGN_DEFAULT_DSS_ENCRYPTIONALGORITHM=RSA ④
DOOKUG_SERVICE_ENGINE_PDF_DIGITALSIGN_DEFAULT_DSS_DIGESTALGORITHM=SHA-256
DOOKUG_SERVICE_ENGINE_PDF_DIGITALSIGN_DEFAULT_PDFBOX_SIGNATUREALGORITHM=SHA256WithRSA ⑤
```

- ① This setting enables the use of the DSS library, which allows placing the image on the PDF.
- ② This is the image that will be added to the document.
- ③ This defines the page where the image will appear in the document.
- ④ In case of USEDSSSIG=true: Depends on the private key, can be ECDSA as well
- ⑤ In case of USEDSSSIG=false: You can configure the private key with this variable. See [documentation](#).

Saxon

```
DOOKUG_SERVICE_ENGINE_SAXON_FOPCONFIG=/home/icellmobilsoft/fop-config/fop-config.xml
DOOKUG_SERVICE_ENGINE_SAXON_XSLT_LANGUAGE_VARIABLE=lang
DOOKUG_SERVICE_ENGINE_SAXON_XSLT_LANGUAGE_DEFAULT=HU
```

Handlebars

```
DOOKUG_SERVICE_ENGINE_HANDLEBARS_HELPER_JAVASCRIPT_DIRECTORY=/home/icellmobilsoft/handlebars/helper/js
DOOKUG_SERVICE_ENGINE_HANDLEBARS_ESCAPINGSTRATEGY=HTML_ENTITY
```

```

CONSOLE_LOGGING_JSON_ENABLED=true
CONSOLE_LOGGING_ENABLED=false

```



If logging in JSON format is required, set the `CONSOLE_LOGGING_ENABLED` environment variable to false to avoid duplicate logs on the console.

1.2. Recommended K8S Configuration

Additional keys can be found here:

- [wildfly/oracle-jdbc configuration](#)
- [wildfly/postgres-jdbc configuration](#)

Parameter	Value	Description
JAVA_OPTS	-Xms2000m -Xmx2000m -XX:MetaspaceSize=96M -XX:MaxMetaspaceSize=256m -XX:+PrintCommandLineFlags -XX:+UseG1GC	Maximize module heap requirement to 2G memory with UseG1GC algorithm.
ORACLE_DS_MAX_POOL_SIZE/ POSTGRESQL_DS_MAX_POOL_SIZE	30	Datasource pool maximum size
OPENTRACING_JAEGER_AGENT_HOST	jaeger	Jaeger host
OPENTRACING_JAEGER_AGENT_PORT	6831	Jaeger agent port
OPENTRACING_JAEGER_AGENT_SAMPLER_PORT	5778	Jaeger agent sampler port
OPENTRACING_JAEGER_PROPAGATION	JAEBER/B3	Jaeger propagation
JAEGER_SERVICE_NAME	dookug-document-service	Service name visible in Jaeger (default: ROOT.war)
LOGSTASH_ENABLED	true	Enable Logstash connector (default false)
LOGSTASH_HOST	tcp:graylog	Logstash server host (do not remove "tcp:" prefix)
LOGSTASH_PORT	12201	Logstash GELF input port
LOGSTASH_K8S_NAMESPACE	dookug-document-service	Set Logstash logging additionalFields K8S_NAMESPACE

LOGGING_K8S_NAMESPACE	dookug-document-service	Value of k8s_namespace key in log line (log pattern)
CONSOLE_LOGGING_ENABLED	false	Disable console logging in Wildfly (default: true)
CONSOLE_LOGGING_JSON_ENABLED	true	Enable JSON logging on console in Wildfly. It is recommended to set CONSOLE_LOGGING_ENABLED environment variable to false, otherwise duplicate logging will occur on the console. Default: false
FILE_LOGGING_ENABLED	false	Disable file logging in Wildfly (default: true)
COFFEE_MODEL_BASE_JAVA_TIME_TIMEZONE_ID	UTC	For OffsetDateTime types to be stored in the database, specify the zone for creation and modification dates. Default is UTC, but can be any valid parameter for <code>java.time.ZoneId.of(String zoneId)</code> method. Incorrect values will generate warning messages and default values will be used, which will be <code>TimeZone.getDefault().toZoneId()</code> .

Requests/limits settings

```
spec:
  containers:
    resources:
      limits:
        cpu: "3"
        memory: 3G
      requests:
        cpu: "3"
        memory: 3G
```

1.3. Observability

1.3.1. Grafana dashboards

The module provides Grafana dashboards that enhance observability and monitoring capabilities.

- `backend_memory_usage.json`:
 - Displays detailed memory usage.
 - Shows data for four different GC algorithms.
- `backend_base.json`:
 - Shows datasource metrics.
 - `pool_available`: Number of available connections.
 - `pool_in_use`: Number of connections currently in use.
 - `blocking_time`: Time spent waiting for a database connection when needed for a service logic operation (e.g., running a `SELECT` or `INSERT`). If no connection is available, this time increases.
 - `pool_timed_out`: Number of active connections that have timed out, for example, during a pause after high system load.

Dashboard tips

- If the `pool_in_use` value consistently reaches the `DS_MAX_POOL_SIZE` limit, it indicates that the pool size is too small or that the CPU core-to-consumer thread count ratio is misconfigured. This may lead to the service handling many messages while simultaneously serving REST endpoints.
- Similarly, if `pool_available` remains at 0 for extended periods, the pool size is insufficient.
- `blocking_time` represents wasted time during which the service could have performed useful tasks. If it is high, it is worth checking whether database operation times have increased or if the pool size is too small.
- `blocking_time` indicates how often the service was unable to acquire a free connection. This is also influenced by the default configuration in `standalone.xml`:

```
<timeout>
  <blocking-timeout-millis>5000</blocking-timeout-millis>
  <idle-timeout-minutes>5</idle-timeout-minutes>
</timeout>
```

The system has 5 seconds to acquire a free connection for a database operation; otherwise, an exception will be thrown.

1.3.2. Health - startup/liveness/readiness

The service supports the use of K8S probes.

Started: <http://localhost:9990/health/started>

Live: <http://localhost:9990/health/live>

Ready: <http://localhost:9990/health/ready>

Additional informations

1. Template Cache

The application caches the TEMPLATE data stored in the database using GUAVA. The caches have a defined lifespan, and the time resets with each new request.



The cache key for a template is its X_ID, which is determined based on the template name, language, and validity. This means that if multiple templates have the same name and the first one expires, the system will automatically switch to the next valid template without being affected by the cache.



If a template's content is updated, you can clear the template cache by calling the /system/evict endpoint. This ensures that the system will use the updated content.

Configuration Parameters:

You can specify the duration for which templates are kept in the cache using the DOOKUG_SERVICE_CACHE_TEMPLATE_TTL parameter, in minutes. The default is 60 minutes.

You can specify whether metrics generation is needed using the DOOKUG_SERVICE_CACHE_TEMPLATE_ENABLESTATISTIC parameter. The default is false, meaning no metrics are generated.

Metrics similar to the following are generated:

```
# TYPE application_cache_hit_count gauge
application_cache_hit_count{name="template"} 0.0
# TYPE application_cache_miss_count gauge
application_cache_miss_count{name="template"} 1.0
# TYPE application_cache_size gauge
application_cache_size{name="template"} 1.0
```

2. Helpers - Helper Functions for Use in Template Files

2.1. Built-in Helpers

Handlebars provides built-in helpers, whose documentation can be found [here](#).

2.2. Custom Helpers

It is possible to use general helpers provided by the module. In the examples, the ... parts render if

the expression evaluates to true.

2.2.1. Usage

When using helpers, specify the parameters for the given helper after the helper keyword. Helpers can be used during inline generation and database-stored template-based generation. The parameters can be hardcoded values or come in JSON format during the document generation call.

Helpers can be combined with each other.

Tok en	Description	Usage	Ava ilab le fro m Ver sio n
equ als	Compares the values of two elements; renders if the evaluation is true.	<pre>{{#if (equals yourField 'white')}} ... {{/if}}</pre>	0.1.0
bef ore	Renders if the provided parameter is an earlier date than the given value. The latter can also be a parameter.	<pre>{{#if (before yourDate '2023-08-13T05:40:55Z')}} ... {{/if}} {{#if (before yourDate checkDate)}} ... {{/if}}</pre>	0.1.0
bet we en	Renders if the provided parameter falls between the two given date values. The latter can also be parameters.	<pre>{{#if (between yourDate '2023-08-13T05:40:55Z' '2023-08-15T05:40:55Z')}} ... {{/if}} {{#if (between yourDate startDate endDate)}} ... {{/if}}</pre>	0.1.0
dat eMi nus Mi nut es	Subtracts a given number of minutes from the specified date.	<pre>{{dateMinusMinutes '2023-08-13T05:40:55Z' 60}} // evaluates to '2023-08-13T04:40:55Z' {{dateMinusMinutes yourDate 60}} // The following two examples evaluate to the same date {{dateMinusMinutes yourDate minutesToSubtract}}</pre>	0.1.0

datePlusMinutes	The counterpart to dateMinusMinutes, it adds minutes to the given date.	<pre> {{datePlusMinutes '2023-08-13T05:40:55Z' 60}} - evaluates to '2023-08-13T06:40:55Z' {{datePlusMinutes yourDate 60}} {{datePlusMinutes yourDate minutesToAdd}} </pre>	0.1.0
declare	Creates a new variable and immediately renders it. The created variables are global, regardless of where they were created in the templates.	<pre> {{declare 'myColor' 'white'}} // immediately renders the word 'white' ... {{#if myColor}} {{myColor}} {{/if}} // renders 'white' </pre>	0.1.0
declareVoid	Works similarly to the declare helper, but does not immediately render the given value. declare can be replaced by declareVoid followed by immediate invocation.	<pre> {{declareVoid 'myColor' 'white'}} ... {{#if myColor}} {{myColor}} {{/if}} // renders 'white' Replacing declare with declareVoid: {{declareVoid 'myColor' 'white'}} {{myColor}} // renders 'white' immediately after creation </pre>	0.1.0
formatDate	Transforms the given parameter, accepted in Java 8 date format, according to the specified pattern. The pattern only accepts date formats; otherwise, it throws an error.	<pre> yourDate = '2024-01-09'; {{formatDate '2023-08-13' 'yyyy.MM.dd'}} // '2023.08.13' {{formatDate '2023-08-13' 'MM.dd.yyyy'}} // '08.13.2023' {{formatDate yourDate 'yyyy/MM/dd'}} // '2024/01/09' </pre>	0.5.0

for mat Dat eTi me	Transforms the given parameter, accepted in Java 8 date and time format, according to the specified pattern. The helper handles full ISO date format input and can generate appropriate output using the pattern. The helper can also handle time zones; if a Java-accepted Zone ID is provided after the output pattern, it converts the input time to the time zone's output.	<pre> yourDate = '2024-01-09T15:30:04Z' yourDateTimeFormat = 'MM.dd-HH:mm' yourDateFormat = 'yyyy/MM.dd' yourTimeFormat = 'hh:mm:ss a' {{formatDateTime '2023-08-13T05:40:55Z' 'yyyy-MM-dd HH:mm:ss'}} // '2023-08-13 06:40:55' {{formatDateTime yourDate 'yyyy-MM-dd HH:mm:ss'}} // '2024-01-09 15:30:04' {{formatDateTime yourDate yourDateTimeFormat}} // '01.09-15:30' {{formatDateTime yourDate yourDateFormat}} // '2024/01.09' {{formatDateTime yourDate yourTimeFormat}} // '03:30:04 PM' {{formatDateTime '2023-08-13T05:40:55Z' 'yyyy-MM-dd HH:mm:ss' 'CET'}} // '2023-08-13 08:40:55' </pre>	0.5. 0
for mat Tim e	Transforms the given parameter, accepted in Java 8 time format, according to the specified pattern. The pattern only accepts time formats; otherwise, it throws an error.	<pre> yourTime = '15:30:55Z'; {{formatTime '15:30:55Z' 'HH:mm:ss'}} // '15:30:55' {{formatTime yourTime 'h:mm A'}} // '3:30 PM' </pre>	0.5. 0

for mat Nu mb er	Helper used for formatting numbers, following Java number formatting conventions.	<pre> number = 1234.567; percentage = 0.4567 {{formatNumber number '#'}} // "1235", rounded integer {{formatNumber number '0.00'}} // "1234.57", number rounded to 2 decimals {{formatNumber number '000000.00'}} // "001234.57", number padded with leading zeros {{formatNumber number '#,###.##'}} // "1,234.57", comma-separated, grouped by thousands {{formatNumber number '\$#,##0.00'}} // "\$1,234.57", currency expression {{formatNumber percentage '0.00%'}} // "45.67%", percentage expression of a value between 0 and 1 {{formatNumber number '0.###E0'}} // "1.235E8", number in scientific notation </pre>	0.1. 0
and	Logical AND operator for N values. Renders if the logical AND operator evaluates to true.	<pre> {{#if (and falseValue trueValue notExistingValue)}} YES {{else}} NO {{/if}} // renders "NO" {{#if (and trueValue trueValue trueValue)}} YES {{/if}} // renders "YES" since all three values are true myValue = 'black' {{#if (and (equals myValue 'black'))}} YES {{else}} NO {{/if}} // renders "YES", both the variable and the string evaluation are true {{#if (and (equals myValue 'white'))}} YES {{else}} NO {{/if}} // renders "NO", both the variable and the string evaluation are false Combined usage: {{#if (and (equals 'black' 'white') (equals 'white' 'white'))}} YES {{else}} NO {{/if}} // renders "NO", the first is false, the second is true </pre>	0.1. 0

or	Logical OR operator for N values. Renders if the logical OR operator evaluates to true.	<pre> {{#if (or falseValue trueValue notExistingValue)}} YES {{else}} NO {{/if}} // renders "YES" {{#if (or falseValue falseValue falseValue)}} YES {{/if}} // renders nothing myValue = 'white' {{#if (or myValue1 myValue2)}} YES {{else}} NO {{/if}} // renders "YES" if myValue1 OR myValue2 is defined (not null) Combined usage: {{#if (or (equals 'black' 'white') (equals 'white' 'white'))}} YES {{else}} NO {{/if}} // renders "YES", false OR true evaluates to true </pre>	0.1. 0
not	Logical NOT operator. Negates the given parameter; renders if the evaluation is true, otherwise does not render.	<pre> {{#if (not falseValue)}} YES {{else}} NO {{/if}} // YES {{#if (not existingValue)}} YES {{else}} NO {{/if}} // NO {{#if (not (equals 'black' 'white'))}} YES {{else}} NO {{/if}} // YES </pre>	0.1. 0
in	Checks if the first parameter matches any of the subsequent elements	<pre> myValue = 'white' {{#if (in myValue 'black' 'gray')}} YES {{else}} NO {{/if}} // NO {{#if (in myValue 'black' 'white' 'gray')}} YES {{else}} NO {{/if}} // YES </pre>	0.1. 0

mat h	<p>Helper for basic mathematical operations.</p> <p>The first parameter is the operator, and the other two parameters are the operands.</p> <p>The list of usable operators:</p> <p>“+”, “-”, “*”, “/”, “%”</p> <p>If an invalid operator is used, the evaluation result is: “-1”. Otherwise, the result of the mathematical operation on the operands corresponding to the operator.</p>	<pre> num1 = 5 num2 = 8 num3 = 100 num4 = 20 {{math '+' num1 num2}} // 13 {{math '-' num3 53}} // 47 {{math '*' num2 num4}} // 160 {{math '/' num3 num4}} // 5 {{math '%' num4 num2}} // 40 {{math 'A' num1 num2}} // -1 </pre>	0.1. 0
----------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	-----------

2.2.2. Additional Helper Functions

The following 3rd party helper functions can be used in the project:

- [StringHelpers](#)

3. Using Custom Fonts

This documentation is intended for the [Apache PDFBox](#) engine and focuses on using custom fonts.



A limitation of the engine is that it can only embed TTF fonts. Additionally, if you want to use weight, styles, and variants settings, it is worth noting that PDFBox cannot always emulate these properly unless you load different variations of the font.

In the docker-compose file, we add the **fonts** directory to our image as `/home/icellmobilsoft/fonts`, so we can access the fonts in this directory locally from the templates.

We have two options for using custom fonts.

1. Load the font directly in the CSS
2. Load the font programmatically, then reference it in the CSS using **font-family**

In the first option, you can specify the font’s location in the CSS within the **@font-face** at-rule using the **src: url()** descriptor.

In this case, internet access will be required during generation, which may not always be possible as the company’s policy may prohibit the module from accessing external URLs.

This can be mitigated either by making the fonts accessible through an internal URL or by preloading some frequently used fonts in the module and referencing their local availability using a **file://** URL. If the module will contain preloaded fonts, it is advisable to mention this in the documentation or provide the information through an endpoint.

TEMPLATE:

```

<style>
  @font-face {
    font-family: 'Cairo';
    font-style: normal;
    src: url(file:/home/icellmobilsoft/fonts/Cairo/Cairo-Regular.ttf); ①
  }

  @font-face {
    font-family: 'IndieFlower';
    font-style: normal;
    src: url(file:fonts/IndieFlower/IndieFlower-Regular.ttf); ②
  }
</style>

```

- ① Loading Cairo-Regular.ttf from the `/home/icellmobilsoft/fonts` directory.
- ② Here we also load the font from the above directory. The server considers `/home/icellmobilsoft` as the root, so the relative path works.

Using Google Fonts



```

<style>
  @import url('https://fonts.googleapis.com/css?family=Quicksand
&amp;display=swap'); ①

  .quicksand { ②
    font-family: 'Quicksand', Arial;
    font-weight: 700;
    font-style: normal;
    font-size: 38px;
    line-height: 1.15;
    letter-spacing: -.02em;
    color: rgba(0, 0, 0, 0.8);
    -webkit-font-smoothing: antialiased;
  }
</style>
...
<p class="quicksand">Google Quicksand Font úőüöóéáyí</p> ③

```

- ① Importing the Quicksand font from [Google Fonts](#)
- ② Setting the CSS class to use the font
- ③ Displaying text with the specified font

If we want to load fonts programmatically from the file system, we can use the `builder.useFonts` method during rendering, which we can reference in the CSS.

JAVA:

```
builder.useFont(new File("fonts/NotoSansThai/NotoSansThai-Regular.ttf"),  
"notosansthai-regular"); ①
```

① Loading NotoSansThai-Regular.ttf

TEMPLATE:

```
<style>  
  @font-face {  
    font-family: 'notosansthai-regular'; ①  
    font-style: normal;  
    -fs-font-subset: complete-font; ②  
  }  
</style>
```

① Using the previously loaded font

② This is just an example; this setting is not really necessary as it embeds the entire font, whereas by default only the subset is embedded, which is the correct operation.

The downside is that all fonts are loaded during rendering, even those not used in the document template.



The PDFBox Fonts Wiki is available [here](#)

4. Digitally Signing PDFs

This documentation is intended for the [Apache PDFBox](#) engine and describes the process of digitally signing PDFs using a self-signed certificate, which is added to the image as `/home/icellmobilsoft/keys/keystore.p12` using the docker-compose file.

The keystore is located in the `/etc/docker-compose/keys` folder and was created on November 7, 2023, with the following command:

```
keytool -genkeypair -storepass 123456 -storetype pkcs12 -alias test -validity 10958 -v  
-keyalg RSA -keystore keystore.p12
```



it will expire on November 7, 2053.

4.1. Configuration

The signature can be requested in the API calls within the GeneratorSetup request. Here you can specify the `signatureName`, `signatureReason`, and the private key alias (`keyAlias`) in case the keystore file contains multiple private keys.

```
...
<ns2:generatorSetup>
  <ns2:generatorEngine>PDF_BOX</ns2:generatorEngine>
  <ns2:templateEngine>NONE</ns2:templateEngine>
  <ns2:responseFormat>PDF</ns2:responseFormat>
  <ns2:digitalSignatureProfile>sampleProfile</ns2:digitalSignatureProfile> ①
  ...

```

① The name of the signature profile only in case you need PDF signing

The signature can be customized with the following configuration keys:

```
dookug:
  service:
    engine:
      pdf:
        digitalsign:
          sampleProfile: ①
          name: Example Ltd. ②
          reason: Certified ②
          keystore: /home/icellmobilsoft/keys/keystore.p12 ③
          keystorePass: 123456 ④
          keystoreType: PKCS12 ④
          keyAlias: key_test ⑤

```

① the name of the signatureProfile

② **name** and **reason** default values are used if the request does not contain these values.

③ **keystore** is mandatory and specifies the location of the signing keys.

④ **keystorePass** and **keystoreType** specify the password and type of the keystore file, respectively. These are also mandatory.

⑤ the identifier of the private key within the keystore

4.2. Changing Certificates

If you need to replace the certificate used for digital signing (e.g., due to expiration), follow these steps:

- mount the new keystore file as a docker volume
- edit the [configurations of digital signing](#) if necessary

No application release is required. Restarting the service with the new certificate is sufficient.

4.3. Visible signature (using the EU DSS ESIGN library)

In case you want to add visible (clickable) signature to the PDF document you need to add some extra configuration.


```

dookug:
  service:
    engine:
      pdf:
        digitalsign:
          sampleProfile:
            keystore: /home/icellmobilsoft/keys/keystore.p12 ①
            keystorePass: 123456
            keystoreType: PKCS12
            keyAlias: key_test
            useEuDssSig: true ②
          dss:
            imageFile: /home/icellmobilsoft/pdfsign/sample/signature.png ③
            showOnPage: 1 ④
            position:
              left: 18 ⑤
              top: 2 ⑥
            width: 2 ⑦
            useTimestamp: true ⑧
            tspSources:
              - http://timestamp.digicert.com ⑨
            useLT: true ⑩
            useLTA: false ⑪
            encryptionAlgorithm: RSA ⑫
            digestAlgorithm: SHA-256 ⑬
            certificationPermission: NO_CHANGE_PERMITTED ⑭

```

- ① the keystore file
- ② use the EU DSS library to sign the document
- ③ the image you want to add to the document
- ④ the image will added to this page: **-1**: last page, **1**:first page, **0**:wont add, **n**:n-th page
- ⑤ X position of the image in cm (from top left corner)
- ⑥ Y position of the image in cm (from top left corner)
- ⑦ the width of the image in cm
- ⑧ you can add the signature timestamp to the image
- ⑨ timestamp server for LT/LTA
- ⑩ for PAdES BASELINE LT
- ⑪ for PAdES BASELINE LTA
- ⑫ encryption algorithm used for private key (for example: RSA, ECDSA)
- ⑬ digest algorithm used for private key
- ⑭ certification permission can be set optionally. The values can be: NO_CHANGE_PERMITTED | MINIMAL_CHANGES_PERMITTED | CHANGES_PERMITTED

The available configuration keys can be found [here](#).



If we need PAdES BASELINE T, LT or LTA, the tspSources must be given!

PAdES BASELINE T: tspSources must be set

PAdES BASELINE LT: tspSources must be set and useLT=true

PAdES BASELINE LTA: tspSources must be set and useLTA=true



If we set the certificationPermission, the PDF document will be certified and not signed in Adobe Acrobat Reader!

4.4. Simple signature (using the PdfBox ESIGN library)

For legacy using there is an option to use the pdfbox engine for simple digital signing.

```
dookug:
  service:
    engine:
      pdf:
        digitalsign:
          sampleProfile:
            keystore: /home/icellmobilsoft/keys/keystore.p12 ①
            keystorePass: 123456
            keystoreType: PKCS12
            keyAlias: key_test
            pdfBox:
              signatureAlgorithm: SHA256WithRSA ②
```

① the keystore file

② the certificate algorithm used for the pdfbox engined digital signing (not DSS!). For example:
SHA256withECDSA Default: SHA256WithRSA



Here you can find the available signature algorithm identifiers

<https://github.com/bcgit/bc-java/blob/main/pkix/src/main/java/org/bouncycastle/operator/DefaultSignatureAlgorithmIdentifierFinder.java#L745>

The available configuration keys can be found [here](#).

5. Keystore Cache for digital signature

The application caches the keystore data configured for signature profiles using GUAVA. The caches have a defined lifespan, and the time resets with each new request.

Configuration Parameters:

You can specify the duration for which templates are kept in the cache using the

DOOKUG_SERVICE_CACHE_KEYSTORE_TTL parameter, in minutes. The default is 1440 minutes (1 day).

You can specify whether metrics generation is needed using the DOOKUG_SERVICE_CACHE_KEYSTORE_ENABLESTATISTIC parameter. The default is false, meaning no metrics are generated.

Metrics similar to the following are generated:

```
# TYPE application_cache_hit_count gauge
application_cache_hit_count{name="keystore"} 0.0
# TYPE application_cache_miss_count gauge
application_cache_miss_count{name="keystore"} 1.0
# TYPE application_cache_size gauge
application_cache_size{name="keystore"} 1.0
```

Release notes

1. v0.4.0

The project has been released on GitHub opensource.

- maven pom change:
 - `<groupId>hu.icellmobilsoft.dookug.common</groupId>` → `<groupId>hu.icellmobilsoft.dookug</groupId>`
 - `<groupId>hu.icellmobilsoft.dookug.document</groupId>` → `<groupId>hu.icellmobilsoft.dookug</groupId>`
 - `<groupId>hu.icellmobilsoft.dookug.client</groupId>` → `<groupId>hu.icellmobilsoft.dookug</groupId>`

2. v0.5.0

2.1. Changes / updates

- Caching of templates in memory <https://github.com/i-Cell-Mobilsoft-Open-Source/DookuG-backend/issues/4>
- Observability: DB health check added
- TemplateLanguage type has been changed from enum to string (EN → EN or en_GB).
- The jandex.idx file has been removed from dookug-client-jee10. It caused Unsatisfied Dependency Exception during the deployment.

2.2. Migration

Replace `TemplateLanguageType.*` occurrences with the string value of the language.

3. v0.6.0

3.1. Changes / updates

- Documentation fixes (formatting, missing 0.5.0 release notes include)
- Add [com.github.jknack.handlebars.helper.StringHelpers](#) build in helper to helpers
- Rename `formatDate` to `formatDateTime`
- Create `formatDate` which handles date format
- Create `formatTime` which handles time format
- Add IT test for owned helpers
- GET `/system/evict` REST endpoint added
- Template caching has been fixed
- Template caching can be configured with the environment variable named `DOOKUG_SERVICE_CACHE_TEMPLATE_ENABLED` (`true` by default)

3.2. Migration

Changes are backwards compatible doesnt need any migration.

4. v1.0.0

4.1. Changes / updates

- Jakarta EE10 upgrade
- Test fix
- gitHub Workflows - java 17 upgrade
- Bugfix: The `not` helper function has been fixed, it evaluated the `false` parameter incorrectly.
- Handlebars version bump: 4.3.1 → 4.4.0
- Handlebars caching: Handlebars template engine now uses its in-built caching mechanism for compiled templates
- coff:ee upgrade - migration doc: [2.5.0 → 2.6.0](#)
- coff:ee upgrade - migration doc: [2.6.0 → 2.7.0](#)



The client is not affected by the coffee upgrade, it remains 2.6.0 coffee . The reason for this is that you should upgrade to 2.7.0 on the project that uses coffee.



Because of the coffee upgrade, the use of interfaces had to be removed from the clients, because it already uses the coffee 2.7.0 BaseException, and would cause a break in the client.

- For `dookug-model`, `org.hibernate` groupId has been replaced by `org.hibernate.orm`.
- Roaster upgrade migration: [2.1.0](#) → [2.2.0](#)
- In tests, the `coffee.model.base.java.time.timezone.id` property in the `BaseIT` class is set to UTC.
- New documentation structure

4.2. Migration

- Replace the `dookug-client` dependency in your project depending on whether you are using JEE 8 or JEE 10.

```
<dependency>
  <groupId>hu.icellmobilsoft.dookug.client</groupId>
  <artifact>dookug-client-jee10</artifact> ①
  <version>1.0.0</version>
</dependency>

<dependency>
  <groupId>hu.icellmobilsoft.dookug.client</groupId>
  <artifact>dookug-client-jee8</artifact> ②
  <version>1.0.0</version>
</dependency>
```

① in case you have EE10 application

② in case you have EE8 application

5. v1.1.0

5.1. Changes / updates

- Documentation changes:
 - it has been translated to english
 - some fixes and changes have been applied also in `openapi`.
- Using the EU DSS Esig library (6.1) to sign PDF documents (optional)
- PDF signing: `signature` key removed from configuration.
- GitHub workflows for docker build and release build
- The `coffee-module-etcd` dependency has been removed.
- The `signatureName` and `signatureReason` has been removed from the request - These are set up in the application configuration under profile name keys

- Possible extSessionId duplication (in client calls) fixed
- SVG image fix
- JsonB context handling fixes
- docker-wildfly base image upgrade 2.0.0 → 2.1.0
 - JSON_MODULE_ID and JSON_MODULE_VERSION envs have been set in Dockerfile.
- TemplatePartContent entity cleanup - unused fields are removed
- Bugfix: The template cache can now be properly cleared by calling the system/evict endpoint
- Local docker compose files are now available under `etc/docker-compose` directory.
- Fixed HandlebarsTemplateCompiler warning on startup
- EC keys can be used with non-DSS digital signing
- CertificationPermission can be set in configuration. It's empty -not used- by default
- new endpoint for PDF signing only

5.2. Migration

Changes are backwards compatible doesn't need any migration.

6. v1.2.0

Accidental release. Does not contain functional changes compared to v1.1.0, only in gitHub ci.

6.1. Changes / updates

- gitHub ci: enabling manual docker release on tags.

6.2. Migration

Changes are backwards compatible doesn't need any migration.

7. v1.3.0

7.1. Changes / updates

- Saxon template caching

7.2. Migration

Changes are backwards compatible doesn't need any migration.