2.1.0 Documentation English



1 Configuration
- 1.1 TI-CA certificates
- 1.2 Environment variables
  - 1.2.1 URLs of the addressed systems
  - 1.2.2 Configurations TIaaS working environment
  - 1.2.3 DIGA Information
  - 1.2.4 ePA Environment
  - 1.2.5 ePA Provider Mapping
  - 1.2.6 API Configuration
  - 1.2.7 Redis configuration
- 1.3 Docker Compose

2 Endpoints
- 2.1 API-Overview
- 2.2 API-Documentation: POST /send-document/
  - 2.2.1 Request Body
  - 2.2.2 Example Request in Python:
  - 2.2.3 Return values /send-document
  - 2.2.4 ErrorCodes /send-document
- 2.3 API-Documentation: POST /localize-record/
  - 2.3.1 Request Body
  - 2.3.2 Example Request in Python
  - 2.3.3 ErrorCodes /localize-record:

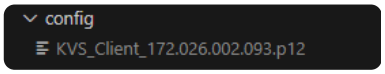
Valid from Version: 2.1.0

This is the documentation for using the “_fbeta ePA Service”. Important configuration details are listed and explained below. The details of the API interface are then described and an example of integration into a production system is given.

Configuration

TI-CA certificates

Place your p12 certificate in the `/config` directory. The certificate is included in operations that communicate with the connector. Only the password for the certificate must be stored as the environment variable `KONNEKTOR_CERT_PW`.



Environment variables

The following environment variables must be set in the .env file:

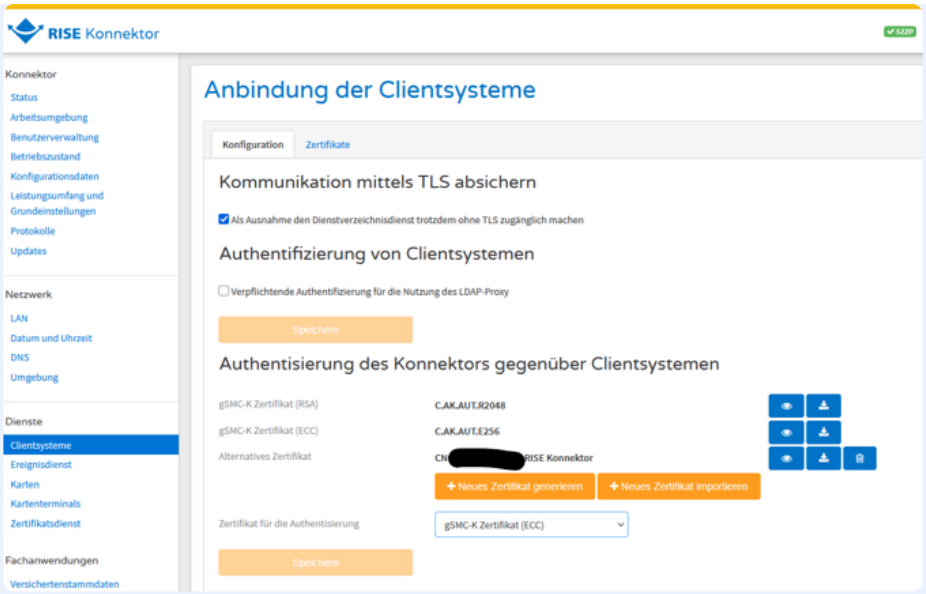
URLs of the addressed systems

- `KONNEKTOR_URL` : URL of your connector. This means: The IP address of your connector in an https:// context.
  (e.g. `'https://173.151.11.5'`)
- `USER_AGENT` : User Agent String for HTTP-Requests
  (`'usrAgent/1.1.0'`)
- `KONNEKTOR_CERT_PW` : Passwort for your p12 certificate
- `KONNEKTOR_TLS_MODE` : (ATTENTION - This env is considered a BREAKING CHANGE):
  - `smc_k` : The TLS handshake is performed using the gSMC-K certificate. This is the recommended mode for production use. The smc_k certificate is issued by the gematik an can be validated by the TI truststore.
  - `alternative` : The TLS handshake is performed using an alternative certificate. This is typically used for testing or development. The alternative certificate is issued by the Konnektor itself and is not validated by the TI truststore. The certificate must be provided in the config/ssl/konnektor/cert.pem file.
  - `insecure` : The TLS handshake is performed without verifying the Konnektor's certificate. This is insecure and NOT recommended for production use.

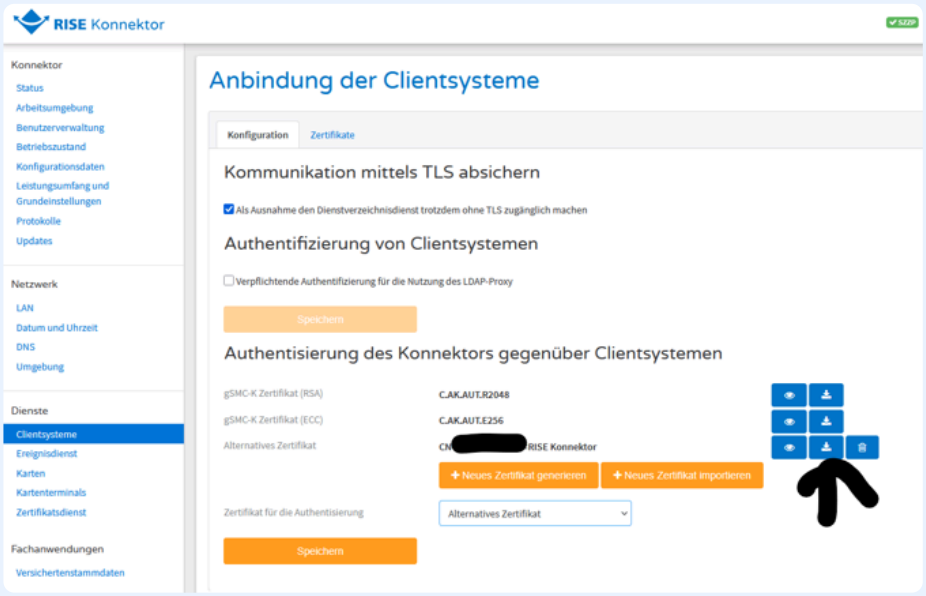


According `KONNEKTOR_TLS_MODE` : Map the mode to the setting configured in the connector.
This means:

- If you choose to use the `smc_k` -Mode, be shure to set `“KonnektorUI > Dienste > Clientsysteme > Konfiguration > Authentisierung des Konnektors gegenüber Clientsystemen > Zertifikat für Authentisierung”` to `“gSMC-K Zertifikat (ECC)”`



- If you choose to use the `alternative`-Mode, be shure to set `“KonnektorUI > Dienste > Clientsysteme > Konfiguration > Authentisierung des Konnektors gegenüber Clientsystemen > Zertifikat für Authentisierung”` to `“Alternatives Zertifikat”`.
  Afterwards download the “Alternatives Zertifikat” and safe it as “cert.pem” in the config-dirctory.



- If you choose to use the `insecure`-Mode, no specific settings are needed.

Configurations TIaaS working environment

- `MANDANT_ID`: ID of the “Mandat” set up in the connector (`'Test'`)
- `CLIENT_SYSTEM_ID`: ID of the “Client-Systems” set up in the connector (`'Test_Clientsystem'`)

- `WORKPLACE_ID` : ID of the “Arbeitsplatz” set up in the connector (`'Test-Arbeitsplatz'`)
- `USER_ID` : ID of the “User” set up in the connector (optional)

DIGA Information

- `DIGA_NAME` : Name of your DiGA (name of the prescription unit) (`'DIGA-Hersteller'`)
- `DIGA_MANUFACTURER` : Name of the DiGA manufacturer (`'Sina Gr..fin ..zTEST-ONLY'`)

ePA Environment

- `EPA_ENVIRONMENT` : ePA-Environment `RU (RU REF) , RT (RU DEV) or PROD`

ePA Provider Mapping

```
RECORD_PROVIDER_1 = 'IBM'
RECORD_PROVIDER_2 = 'Bitmarck Technik'
RECORD_PROVIDER_X = 'PROVIDERNAME'[...]
```

API Configuration

- `UVICORN_WORKERS` : Number of API workers for parallel processing (default value: 1)
- `API_SSL_CERT` : Path to the TLS certificate (in the container) used by the API (optional)
- `API_SSL_KEY` : Path to the corresponding private key (in the container) used by the API (optional)

Redis configuration

Base configuration

- `REDIS_HOST` : Redis server host (default: `redis`)
- `REDIS_PORT` : Redis server port (default: `6379`)
- `REDIS_PASSWORD` : Redis server password (optional)
- `REDIS_DB` : Redis database number (default: `0`)

TLS configuration

- `REDIS_SSL_ENABLED` (`true` / `false`): Enables TLS for the Redis connection (default: `false`)
- `REDIS_SSL_VERIFY_CERTS` (`true` / `false`): Enables validation of Redis server certificates (default: `true`)
- `REDIS_SSL_CA_CERT_PATH` : Path to the CA file (in the container) for Redis server certificate validation (required when `REDIS_SSL_VERIFY_CERTS=true` and a private/self-signed CA is used)

mTLS configuration

- `REDIS_MTLS_CERT_PATH`: Path to the Redis client certificate (in the container) for mTLS
- `REDIS_MTLS_KEY_PATH`: Path to the Redis client key (in the container) for mTLS

Docker Compose

Your config directory containing the necessary configuration files needs to be mounted into the Docker container to `/app/config`:

```yaml
volumes:
  - ./config:/app/config
```

To enable TLS for the API endpoints, mount the SSL certificates into the container (if not already mounted through the `config` mount) and set the environment variables (`API_SSL_CERT` and `API_SSL_KEY`) to the path inside the container, e.g.:

```yaml
volumes:
  - ./config/api-cert.pem:/app/config/api-cert.pem:ro
  - ./config/api-key.pem:/app/config/api-key.pem:ro
environment:
  API_SSL_CERT: /app/config/api-cert.pem
  API_SSL_KEY: /app/config/api-key.pem
```

To connect to Redis using TLS or mTLS, mount any required certificates into the container and set the environment variables (`REDIS_SSL_CA_CERT_PATH`, `REDIS_MTLS_CERT_PATH`, `REDIS_MTLS_KEY_PATH`) to the paths inside the container, e.g.:

```yaml
volumes:
  - ./config/ca-cert.pem:/app/config/ca-cert.pem:ro
  - ./config/client-cert.pem:/app/config/client-cert.pem:ro
  - ./config/client-key.pem:/app/config/client-key.pem:ro
environment:
  REDIS_SSL_CA_CERT_PATH: /app/config/ca-cert.pem
  REDIS_MTLS_CERT_PATH: /app/config/client-cert.pem
  REDIS_MTLS_KEY_PATH: /app/config/client-key.pem
```

Endpoints

API-Overview

An overview of all API endpoints and parameters can be obtained via the following endpoints in the container (“Swagger”):

- `/redoc`
- `/docs`
- `/openapi.json`

API-Documentation: `POST /send-document/`

This endpoint is used to send a medical document (MIO) and PDF-A data to the ePA. The request contains the KVNR (insurant_id) of the insured person, a SubmissionSet entry, a

document entry and the XML data, or PDF-A data of the document.

`POST http://localhost:8000/send-document/`

Request Body

The request must be sent in JSON format. The fields in the payload are as follows:

Fieldname	Typ	**Meaning**
`metadata.insurantId`	`string`	KVNR (insurant_id) of the insured person
`metadata.submissionSet`	`object`	Objekt mit Metadaten zum Submission Set
`metadata.submissionSet.entryUUID`	`string`	Unique UUID of the Submission Set
`metadata.documentEntry`	`object`	Object with metadata to the actual document
`metadata.documentEntry.creationTime`	`string`	Time of creation of the document in the format `"%Y%m%d%H%M%S"` (UTC) (e.g: 20250327165707)
`metadata.documentEntry.title`	`string`	Title of the document
`dmetadata.documentEntry.URI`	`string`	location of the file (e.g. `testmio.xml`)
`metadata.documentEntry.entryUUID`	`string`	Unique UUID for the document This UUID should be stored by the DiGA in order to perform a "replace" operation if necessary.
`metadata.documentEntry.old_entry_uuid`	`string` **(optional)**	Unique UUID for the document to be exchanged **(optional)**

`file`	`string` /bytes	Als Bytes gelesener Inhalt des Dokuments

Example Request in Python:

```python
import json
import os
import requests
import uuid
import time

# Define the path to the file to be uploaded
xml_path = os.path.join(os.path.dirname(__file__), '..', 'data', 'REAL_

# Generate UUIDs - They should also be safed
document_uuid = str(uuid.uuid4())
submission_set_uuid = str(uuid.uuid4())

# Get file creation time from metadata
file_creation_time = os.path.getctime(xml_path)
adjusted_time = time.localtime(file_creation_time)
creation_time = time.strftime("%Y%m%d%H%M%S", adjusted_time)

url = "http://localhost:8000/send-document/"
payload = {
    "insurantId": "X110580673", # Insert a valid KVNR (insurant) here
    # See: https://gemspec.gematik.de/docs/gemSpec/gemSpec_Aktensystem_
    "providerId": 2, # Insert the assigned number of the ePA provider o
    "submissionSet": {
        "entryUUID": submission_set_uuid
    },
    "documentEntry": {
        # UTC
        "creationTime": creation_time,
        "title": "Testdokument",
        "URI": "REAL_EXAMPLE_1_KBV_PR_MIO_DIGA_Bundle.xml",
        "entryUUID": document_uuid,
        # OPTIONAL - If you want to replace a document
        # "oldEntryUUID": "929850b6-a5f2-46ca-b742-1abb754cfd45"
    },
}

print("Payload:", json.dumps(payload, indent=4))

# Send the request with 120 seconds timeout
response = requests.post(url,
    data={
        "metadata": json.dumps(payload),
    },
    files={
        "file": open(xml_path, "rb"),
    },
    timeout=120
)

response_dict = json.loads(response.text)
print("Response: ", json.dumps(response_dict, indent=4))
```

Return values /send-document

Code	**Meaning**	Response-Body (JSON)
200	OK – Document processed successfully	• `success` : Boolean. Indicates whether the upload was successful or not. • `message` : String. Summary of the patient record's response to the upload attempt. • `details` : Object. The exact response of the patient record to the upload attempt. • `submissionSetEntryUUID` : String. The entryUUID of the upload attempt (SubmissionSet). • `documentEntryEntryUUID` : String. The entryUUID of the document sent (DocumentEntry).
422	Unprocessable Content – Incorrect or missing request parameters in the JSON payload	• `detail` : List of objects. Contains notes on missing or incorrectly used fields in the request. ◦ `type` : String. ◦ `loc` : Object. ◦ `msg` : String. ◦ `input` : Any.
400	Bad Request – Invalid Payload	• `detail` : ◦ `error` : String. Short description of the error. ◦ `code` : String. The name of the error code. ◦ `type` : String. The name of the component in which the error occurred. ◦ `entryUUID` : String. The submissionSet.EntryUUID of the upload attempt (for debugging purposes).

500	Internal Server Error	◦ `*detail`: Dict<String>. If an error is due to an unsuccessful response from the case system, the specific response from the case system will be displayed here. These details are for troubleshooting purposes only and have no programmatic use.

ErrorCodes /send-document

If an error occurs before the document can be sent to the patient record, the service responds with one of the following ErrorCodes:

ErrorCode	Meaning	Responsibility / Solution
`VAU_INIT_FAILED`	Initialization of the VAU channel between client and server failed	• DiGA: ◦ Connection between ePA API and connector faulty. ◦ SMC-B not accessible via connector/ SMC-B not verified. • File system operator/ Gematik: Malfunctions in the TI or file system.
`VAU_CERT_VALIDATION_FAILED`	Validation of the VAU server certificate not possible	• File system operator/ Gematik: Malfunctions in the TI or file system.
`VAU_AUTH_FAILED`	Authentication via the VAU channel failed (invalid or expired token)	• DiGA: Error in Redis, token is not updated. • File system operator/Gematik: Malfunctions in the TI or file system

`VAU_COMMUNICATION_ERROR`	General error when sending or processing VAU messages	• DiGA: ◦ Connection between ePA API and connector faulty. ◦ SMC-B not accessible via connector/ SMC-B not verified. • File system operator/ Gematik: Malfunctions in the TI or file system.
`EPA_GETNONCE_ERROR`	Error when retrieving a nonce from the ePA authorization service	• DiGA: ◦ Connection between ePA API and connector faulty. ◦ SMC-B not accessible via connector/ SMC-B not verified. • File system operator/ Gematik: Malfunctions in the TI or file system.
`EPA_AUTHZ_ERROR`	Authorization with the ePA service failed	• DiGA: ◦ Connection between ePA API and connector faulty. ◦ SMC-B not accessible via connector/ SMC-B not verified. • File system operator/ Gematik: Malfunctions in the TI or file system.
`EPA_NOT_ENTITLED`	User has not enabled access to the ePA via the FdV app	• User: must approve the DiGA in the FdV app for writing to the ePA.

`EPA_SEND_ERROR`	Error when sending a document to the ePA system	
`KONNEKTOR_INIT_FAILED`	Initialization of the connection to the connector failed	• DiGA: ◦ Connection between ePA API and connector is faulty.
`KONNEKTOR_REQUEST_FAILED`	Request to the connector failed	• DiGA: ◦ Connection between ePA API and connector is faulty.
`KONNEKTOR_MALFORMED_REQUEST`	Request sent to the connector was incorrect	• DiGA: ◦ Incorrect input parameters
`CARD_NOT_FOUND`	The SMC-B card was not found in the card terminal	• DiGA: ◦ SMC-B not accessible via connector
`CARD_CERTIFICATE_ERROR`	Error when retrieving or processing the card certificate	• DiGA: ◦ SMC-B not accessible via connector/ SMC-B not verified
`CARD_PIN_NOT_VERIFIED`	The PIN of the SMC-B card is not verified/activated	• DiGA: ◦ SMC-B not verified
`CARD_OPERATION_ERROR`	An error occurred in the connector during a card operation.	• DiGA: ◦ Check SMC-B manually ▪ Is SMC-B plugged in? ▪ Is SMC-B being read correctly via GetCards?

		▪ Is SMC-B displayed as verified via getPINStatus? • Connector operator: Malfunction in the TI or on the connector
`DOC_DECODE_ERROR`	Error when decoding the document data	• File system operator/Gematik: Disruptions in the TI or file system
`DOC_MISSING_DATA`	Required data for the document is missing	• DiGA: Erforderliche Daten für das Dokument fehlen (URI).
`DOC_UPLOAD_ERROR`	Error uploading the document to the ePA system	• DiGA: ◦ Connection between ePA API and connector faulty. ◦ SMC-B not accessible via connector/ SMC-B not verified. ◦ Error in the data format of the uploaded document. • File system operator/ Gematik: Malfunctions in the TI or file system.
`DOC_UPLOAD_DUPLICATE`	Document already exists in the KVNR electronic patient file. (File hash already in the file system)	• DiGA: MIO or PDF-A has not been modified since the last upload, or it has been accidentally uploaded again.
`DOC_REPLACE_UUID_ERROR`	The document could not be	• DiGA:

	overwritten or replaced because the specified oldEntryUUID does not correspond to any entryUUID in the file system of the relevant KVNR.	◦ The entryUUID of the document to be replaced was not correctly saved, repeated, or set.
`IDP_VERIFICATION_ERROR`	Error when verifying tokens or signatures from the identity provider	• Gematik: Disruptions in the TI or at the central IDP
`IDP_REQUEST_ERROR`	Request to the identity provider failed	• DiGA: Connection between ePA API and IDP faulty. • Gematik: Malfunctions in the TI or at the central IDP
`SSL_ERROR`	An error occurred during a certificate validation.	• DiGA: ◦ Error retrieving certificates. (Check logs)
`UNEXPECTED_ERROR`	An unexpected error that does not fall into other categories	

API-Documentation: `POST /localize-record/`

The endpoint systematically searches all configured record system providers (e.g. the various health insurance systems) to find out which provider stores the electronic patient record of a particular insured person. The result is cached to optimize repeated requests.

`POST http://localhost:8000/localize-record/`

Request Body

The request must be sent in JSON format. The fields in the payload are as follows:

Fieldname	Type	Meaning
`insurantId`	`string`	The insurance number (KVNR) of the patient whose file is to be localized
`ignoreCache`	`bool`	Boolean flag whether the cache should be ignored (optional, default: `false`)

Example Request in Python

```python
url = "http://localhost:8000/localize-record/"

# Send the request with 120 seconds timeout
payload = {
    "insurantId": "Z987654321",
    # "ignoreCache": True
}
response = requests.post(url, json=payload, timeout=120)
```

ErrorCodes /localize-record:

StatusCode	Meaning	Responsibility / Solution
`RECORD_EXISTS_AND_ACTIVATED`	KVNR (insurants id) is registered and activated with one of the file system operators	-
`RECORD_NOT_FOUND`	KVNR (insurants id) was not found in any of the file system operators.	User: must register with KVNR (insurants id) at a file system operator.
`RECORD_STATUS_CONFLICT`	KVNR (insurants id) was found at a file system operator, but does not appear to be	User: must have their access reactivated by the registered file system operator.

	activated (ePA is deactivated)	