2.1.0 Dokumentation



1 Konfiguration
- 1.1 TI-CA Zertifikate
- 1.2 Umgebungsvariablen
  - 1.2.1 URLs der adressierten Systeme
  - 1.2.2 Konfigurationen TIaaS Arbeitsumgebung
  - 1.2.3 DIGA Informationen
  - 1.2.4 ePA Umgebung
  - 1.2.5 ePA Provider Mapping
  - 1.2.6 API-Konfiguration
  - 1.2.7 Redis-Konfiguration
- 1.3 Docker Compose

2 Endpunkte
- 2.1 API-Übersicht:
- 2.2 API-Dokumentation: POST /send-document/
  - 2.2.1 Request Body
  - 2.2.2 Beispiel Request in Python:
  - 2.2.3 Rückgabewerte /send-document
  - 2.2.4 ErrorCodes /send-document
- 2.3 API-Dokumentation: POST /localize-record/
  - 2.3.1 Request Body
  - 2.3.2 Beispiel Request in Python
  - 2.3.3 ErrorCodes /localize-record:

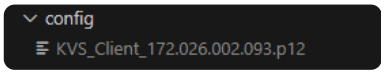
Gültig ab Version: 2.1.0

Dies ist die Dokumentation für die Verwendung des “**_fbeta ePA Service**”. Im Folgenden werden wichtige Konfigurationsdetails aufgeführt und erläutert. Anschließend werden die Details der API-Schnittstelle beschrieben und ein Beispiel für die Integration in ein Produktivsystem gegeben.

Konfiguration

TI-CA Zertifikate

Legen Sie Ihr p12-Zertifikat in das Verzeichnis `/config` ab. Das Zertifikat wird bei Operationen die mit dem Konnektor kommunizieren mit einbezogen. Lediglich das Passwort für das Zertifikat muss als Umgebungsvariable `KONNEKTOR_CERT_PW` hinterlegt werden.



Umgebungsvariablen

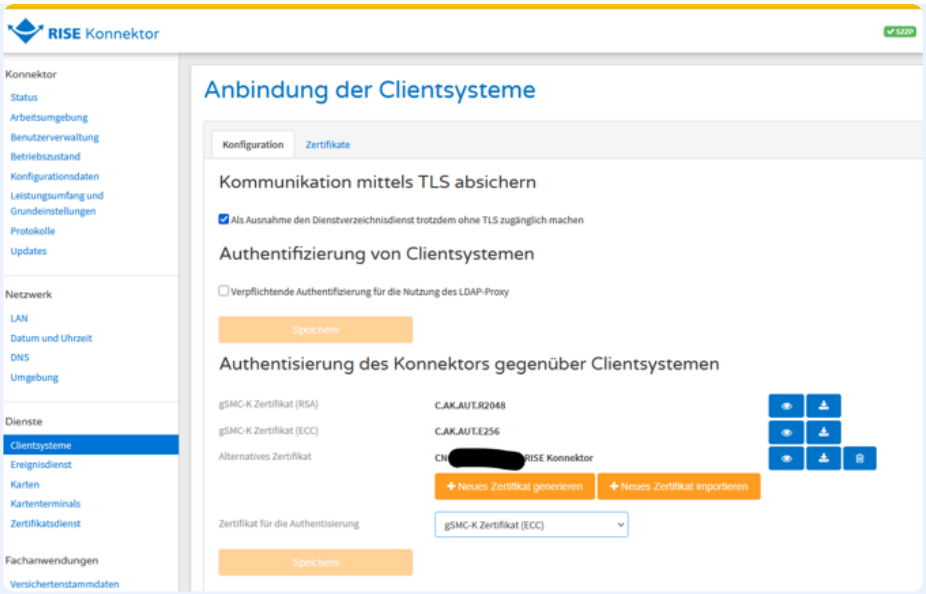
Folgende Umgebungsvariablen müssen in der Datei .env gesetzt werden:

URLs der adressierten Systeme

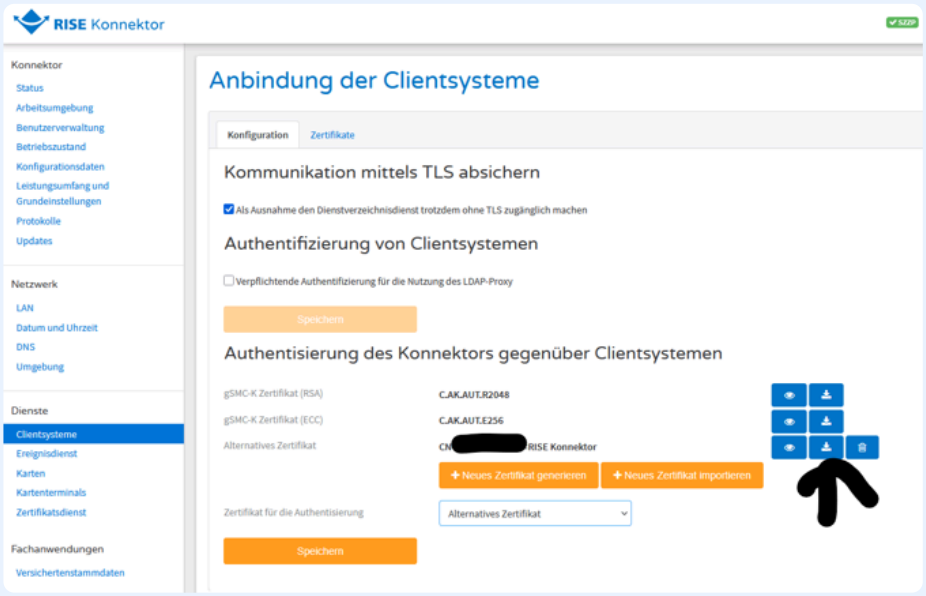
- `KONNEKTOR_URL` : URL Ihres Konnektors. Das bedeutet: Die IP-Address Ihres Konnektors in einem HTTPS-Kontext.
  (bspw. `'https://173.151.11.5'`)
- `USER_AGENT` : User Agent String für HTTP-Requests
  (`'usrAgent/1.1.0'`)
- `KONNEKTOR_CERT_PW` : Passwort für das p12 Zertifikat
- `KONNEKTOR_TLS_MODE` : (ACHTUNG – Diese Umgebungsvariable wird als BREAKING CHANGE betrachtet):
  - `smc_k` : Der TLS-Handshake wird mit dem gSMC-K-Zertifikat durchgeführt. Dies ist der empfohlene Modus für den Produktivbetrieb. Das smc_k-Zertifikat wird von der gematik ausgestellt und kann über den TI-Truststore validiert werden.
  - `alternative` : Der TLS-Handshake wird mit einem alternativen Zertifikat durchgeführt. Dies wird typischerweise für Test- oder Entwicklungszwecke verwendet. Das alternative Zertifikat wird vom Konnektor selbst ausgestellt und wird nicht über den TI-Truststore validiert. Das Zertifikat muss in der Datei config/ssl/konnektor/cert.pem bereitgestellt werden.
  - `insecure` : Der TLS-Handshake wird ohne Überprüfung des Konnektor-Zertifikats durchgeführt. Dies ist unsicher und wird für den Produktivbetrieb NICHT empfohlen.

Bezüglich `KONNEKTOR_TLS_MODE` : Wählen Sie den gewählten Modus so, dass es der Konfiguration des Konnektors entspricht.
Das bedeutet:

- Wenn Sie den `smc_k` -Modus gewählt haben, gehen Sie sicher dass `“KonnektorUI > Dienste > Clientsysteme > Konfiguration > Authentisierung des Konnektors gegenüber Clientsystemen > Zertifikat für Authentisierung”` auf `“gSMC-K Zertifikat (ECC)”` gesetzt ist.



- Wenn Sie den `alternative`-Modus gewählt haben, gehen Sie sicher dass `“KonnektorUI > Dienste > Clientsysteme > Konfiguration > Authentisierung des Konnektors gegenüber Clientsystemen > Zertifikat für Authentisierung”` aiuf `“Alternatives Zertifikat”` gesetzt ist.
  Laden Sie danach das “Alternatives Zertifikat” runter und speichern es als “cert.pem” im config-Ordner.



- Wenn Sie den `insecure`-Modus gewählt haben, sind keine weiteren Einstellungen nötig.

Konfigurationen TIaaS Arbeitsumgebung

- `MANDANT_ID`: ID des eingerichteten Mandanten im Konnektor (`'Test'`)

- `CLIENT_SYSTEM_ID`: ID des eingerichteten Client-Systems im Konnektor (`'Test_Clientsystem'`)
- `WORKPLACE_ID`: ID des eingerichteten Arbeitsplatzes im Konnektor (`'Test-Arbeitsplatz'`)
- `USER_ID`: ID des eingerichteten Users (Optional)

DIGA Informationen

- `DIGA_NAME`: Name Ihrer DiGA (Name der Verordnungseinheit) (`'DIGA-Hersteller'`)
- `DIGA_MANUFACTURER`: Name des DiGA-Herstellers (`'Sina Gr..fin ..zTEST-ONLY'`)

ePA Umgebung

- `EPA_ENVIRONMENT`: ePA-Umgebung `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-Konfiguration

- `UVICORN_WORKERS`: Anzahl der API-Worker für eine parallelisierte Verarbeitung (Default-Wert: 1)
- `API_SSL_CERT`: Pfad zum TLS-Zertifikat (im Container), das von der API verwendet wird (optional)
- `API_SSL_KEY`: Pfad zum dazugehörigen privaten Schlüssel (im Container), der von der API verwendet wird (optional)

Redis-Konfiguration

Basis-Konfiguration

- `REDIS_HOST`: Redis-Server-Host (Standard: `redis`)
- `REDIS_PORT`: Redis-Server-Port (Standard: `6379`)
- `REDIS_PASSWORD`: Redis-Server-Passwort (optional)
- `REDIS_DB`: Redis-Datenbank-Nummer (Standard: `0`)

TLS-Konfiguration

- `REDIS_SSL_ENABLED` (`true` / `false`): Aktiviert TLS für die Redis-Verbindung (Standard: `false`)

- `REDIS_SSL_VERIFY_CERTS` (`true` / `false`): Aktiviert die Prüfung der Redis-Server-Zertifikate (Standard: `true`)
- `REDIS_SSL_CA_CERT_PATH` : Pfad zur CA-Datei (im Container) für die Redis-Server-Zertifikatsprüfung (erforderlich bei `REDIS_SSL_VERIFY_CERTS=true` und privater bzw. selbst-signierter CA)

mTLS-Konfiguration

- `REDIS_MTLS_CERT_PATH` : Pfad zum Redis-Client-Zertifikat (im Container) für mTLS
- `REDIS_MTLS_KEY_PATH` : Pfad zum Redis-Client-Key (im Container) für mTLS

Docker Compose

Das Konfigurationsverzeichnis mit den benötigten Konfigurationsdateien muss in den Docker-Container nach `/app/config` gemountet werden:

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

Um TLS für die API-Endpunkte zu aktivieren, mounten Sie die SSL-Zertifikate in den Container (falls nicht bereits über den `config` -Mount eingebunden) und setzen Sie die Umgebungsvariablen (`API_SSL_CERT` und `API_SSL_KEY`) auf den Pfad im Container, z. B.:

```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
```

Um Redis mit TLS oder mTLS zu verbinden, mounten Sie alle erforderlichen Zertifikate in den Container und setzen Sie die Umgebungsvariablen (`REDIS_SSL_CA_CERT_PATH` , `REDIS_MTLS_CERT_PATH` , `REDIS_MTLS_KEY_PATH`) auf die Pfade im Container, z. B.:

```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
```

Endpunkte

API-Übersicht:

Eine Übersicht aller API-Endpunkte, sowie Parameter sind über folgende Endpunkte im Container zu erhalten (“Swagger”):

- `/redoc`

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

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

Dieser Endpunkt dient zum Senden eines medizinischen Dokuments (MIO) an die ePA. Die Anfrage enthält die KVNR (insurent_id) der versicherten Person, einen SubmissionSet-Eintrag, einen Dokumenteneintrag sowie die XML-Daten des Dokuments.

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

Request Body

Der Request muss im JSON-Format gesendet werden. Die Felder im Payload sind wie folgt:

Feldname	Typ	Beschreibung
`metadata.insurantId`	`string`	KVNR der versicherten Person
`metadata.submissionSet`	`object`	Objekt mit Metadaten zum Submission Set
`metadata.submissionSet.entryUUID`	`string`	Eindeutige UUID des Submission Sets
`metadata.documentEntry`	`object`	Objekt mit Metadaten zum eigentlichen Dokument
`metadata.documentEntry.creationTime`	`string`	Erstellungszeitpunkt des Dokuments im Format `"%Y%m%d%H%M%S"` (UTC) (z.B: 20250327165707)
`metadata.documentEntry.title`	`string`	Titel des Dokuments
`metadata.documentEntry.URI`	`string`	Speicherort oder Name der Datei (z. B. `testmio.xml`)
`metadata.documentEntry.entryUUID`	`string`	Eindeutige UUID für das Dokument

		Diese UUID sollte von der DiGA gespeichert werden um ggf. eine “replace”-Operation durchzuführen.
`metadata.documentEntry.oldEntryUUID`	`string` **(optional)**	Eindeutige UUID für das Dokument, welches ausgetauscht werden soll **(optional)**
`file`	`string` / bytes	Als Bytes gelesener Inhalt des Dokuments

Beispiel 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)
# Transform into UTC-Time
adjusted_time = time.gmtime(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 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))
```

Rückgabewerte /send-document

Code	Bedeutung	Response-Body (JSON)
200	OK – Dokument erfolgreich verarbeitet und verschickt	• `success` : *Boolean*. Gibt an, ob der Upload erfolgreich war oder nicht. • `message` : *String*. Zusammenfassung der Antwort der Patientenakte auf den Upload-Versuch. • `details` : *Objekt*. Die genaue Antwort der Patientenakte auf den Uploadversuch. • `submissionSetEntryUUID` : *String*. Die `entryUUID` des Upload-Versuchs (SubmissionSet). • `documentEntryEntryUUID` : *String*. Die `entryUUID` des verschickten Dokuments (DocumentEntry).
422	Unprocessable Content – Falsche oder fehlende Request-Parameter im JSON-Payload	• `detail` : *Liste an Objekten*. Enthält Hinweise zu fehlenden oder falsch verwendeten Feldern in der Anfrage. ◦ `type` : *String*. ◦ `loc` : *Objekt*. ◦ `msg` : *String*. ◦ `input` : *Any*.
400+	Bad Request – Ungültiger Payload oder fehlende Felder	• `detail` : ◦ `error` : *String*. Kurze Beschreibung des Fehlers. ◦ `code` : *String*. Der Name des **ErrorCodes**. ◦ `type` : *String*. Der Name der Komponente, in dem der Fehler aufgetreten ist.

		◦ `entryUUID`: *String*. Die `submissionSet.EntryUUID` des Upload-Versuchs (zu debugging Zwecken). ◦ `*detail`: Dict<String>. Ist ein Fehler auf eine nicht erfolgreiche Antwort des Aktensystems zurückzuführen, wird der konkrete Response des Aktensystems an dieser Stelle angezeigt. Diese Informationen dienen ausschließlich der Unterstützung beim Troubleshooting und haben keinen programmatischen Nutzen.
500+	Internal Server Error – Fehler bei der Verarbeitung auf Serverseite	

ErrorCodes /send-document

Tritt ein Fehler auf, bevor das Dokument an die Patientenakte gesendet werden kann, antwortet der Service mit einem der folgenden ErrorCodes:

ErrorCode	Bedeutung	Verantwortung / Lösung
`VAU_INIT_FAILED`	Initialisierung des VAU-Kanals zwischen Client und Server fehlgeschlagen	• DiGA: ◦ Verbindung zwischen ePA-API und Konnektor fehlerhaft. ◦ SMC-B nicht über Konnektor abrufbar/ SMC-B nicht verifiziert • Aktensystembetreiber/ Gematik: Störungen in der TI oder am Aktensystem
`VAU_CERT_VALIDATION_FAILED`	Validierung des VAU-Server-Zertifikats nicht möglich	• Aktensystembetreiber/ Gematik: Störungen in der TI oder am Aktensystem
`VAU_AUTH_FAILED`	Authentifizierung über den VAU-Kanal fehlgeschlagen	• DiGA: Fehler im Redis, Token wird nicht aktualisiert.

	(ungültiges oder abgelaufenes Token)	• Aktensystembetreiber/ Gematik: Störungen in der TI oder am Aktensystem
`VAU_COMMUNICATION_ERROR`	Allgemeiner Fehler beim Senden oder Verarbeiten von VAU-Nachrichten	• DiGA: ◦ Verbindung zwischen ePA-API und Konnektor fehlerhaft. ◦ SMC-B nicht über Konnektor abrufbar/ SMC-B nicht verifiziert • Aktensystembetreiber/ Gematik: Störungen in der TI oder am Aktensystem
`EPA_GETNONCE_ERROR`	Fehler beim Abrufen eines Nonce vom ePA-Autorisierungsservice	• DiGA: ◦ Verbindung zwischen ePA-API und Konnektor fehlerhaft. ◦ SMC-B nicht über Konnektor abrufbar/ SMC-B nicht verifiziert • Aktensystembetreiber/ Gematik: Störungen in der TI oder am Aktensystem
`EPA_AUTHZ_ERROR`	Autorisierung mit dem ePA-Dienst fehlgeschlagen	• DiGA: ◦ Verbindung zwischen ePA-API und Konnektor fehlerhaft. ◦ SMC-B nicht über Konnektor abrufbar/ SMC-B nicht verifiziert • Aktensystembetreiber/ Gematik: Störungen in der TI oder am Aktensystem

`EPA_NOT_ENTITLED`	Benutzer hat den Zugriff auf die ePA nicht über die FdV-App freigegebe	User: muss die DiGA in der FdV-App zum schreiben auf die ePA freigeben.
`EPA_SEND_ERROR`	Fehler beim Senden eines Dokuments an das ePA-System	
`KONNEKTOR_INIT_FAILED`	Initialisierung der Verbindung zum Konnektor fehlgeschlagen	• DiGA: ◦ Verbindung zwischen ePA-API und Konnektor fehlerhaft.
`KONNEKTOR_REQUEST_FAILED`	Anfrage an den Konnektor fehlgeschlagen	• DiGA: ◦ Verbindung zwischen ePA-API und Konnektor fehlerhaft.
`KONNEKTOR_MALFORMED_REQUEST`	An den Konnektor gesendete Anfrage war fehlerhaft	• DiGA: ◦ Fehlerhafte Eingabeparameter
`CARD_NOT_FOUND`	Die SMC-B-Karte wurde nicht im Kartenterminal gefunden	• DiGA: ◦ SMC-B nicht über Konnektor abrufbar
`CARD_CERTIFICATE_ERROR`	Fehler beim Abrufen oder Verarbeiten des Karten-Zertifikats	• DiGA: ◦ SMC-B nicht über Konnektor abrufbar/ SMC-B nicht verifiziert
`CARD_PIN_NOT_VERIFIED`	Die PIN der SMC-B-Karte ist nicht verifiziert/aktiviert	• DiGA: ◦ SMC-B nicht verifiziert

`CARD_OPERATION_ERROR`	Während einer Kartenoperation ist ein Fehler im Konnektor aufgetreten	• DiGA: ◦ SMC-B händisch prüfen ▪ Steckt SMC-B ▪ Wird SMC-B über GetCards richtig gelesen ▪ Wird SMC-B über getPINStatus als verifiziert angezeigt. • Konnektorbetreiber: Störung in der TI oder am Konnektor
`DOC_DECODE_ERROR`	Fehler beim Dekodieren der Dokumentdaten	• Aktensystembetreiber/ Gematik: Störungen in der TI oder am Aktensystem
`DOC_MISSING_DATA`	Erforderliche Daten für das Dokument fehlen	• DiGA: Erforderliche Daten für das Dokument fehlen (URI).
`DOC_UPLOAD_DUPLICATE`	Dokument besteht bereit in der elektronischen Patientenakte der KVNR. (File-Hash bereits im Aktensystem)	• DiGA: MIO oder PDF-A wurde seit dem letzten Upload nicht angepasst, bzw. es wird versehentlich erneut hochgeladen.
`DOC_UPLOAD_ERROR`	Fehler beim Hochladen des Dokuments in das ePA-System	• DiGA: ◦ Verbindung zwischen ePA-API und Konnektor fehlerhaft. ◦ SMC-B nicht über Konnektor abrufbar/ SMC-B nicht verifiziert

		◦ Fehler im Datenformats des Uploaddokuments. • Aktensystembetreiber/ Gematik: Störungen in der TI oder am Aktensystem
`DOC_REPLACE_UUID_ERROR`	Das Dokument konnte nicht überschrieben bzw. ersetzt werden, da die angegebene **oldEntryUUID** keiner im Aktensystem der betreffenden KVNR vorhandenen **entryUUID** entspricht.	• DiGA: ◦ entryUUID des zu ersetzenden Dokuments wurde nicht korrekt gesichert, wiedergeholt oder gesetzt.
`IDP_VERIFICATION_ERROR`	Fehler bei der Verifizierung von Tokens oder Signaturen vom Identity Provider	• Gematik: Störungen in der TI oder am zentralen IDP
`IDP_REQUEST_ERROR`	Anfrage an den Identity Provider fehlgeschlagen	• DiGA: Verbindung zwischen ePA-API und IDP fehlerhaft. • Gematik: Störungen in der TI oder am zentralen IDP
`SSL_ERROR`	Bei einer Zertifikatsprüfung ist ein Fehler aufgetreten.	• DiGA: ◦ Fehler beim beziehen der Zertifikate. (In Logs schauen)
`UNEXPECTED_ERROR`	Ein unerwarteter Fehler, der nicht in	-

	andere Kategorien fällt	

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

Der Endpunkt durchsucht systematisch alle konfigurierten Aktensystem-Provider (z.B. die verschiedenen Krankenkassen-Systeme), um herauszufinden, bei welchem Anbieter die elektronische Patientenakte eines bestimmten Versicherten gespeichert ist. Das Ergebnis wird cached, um wiederholte Anfragen zu optimieren.

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

Request Body

Der Request muss im JSON-Format gesendet werden. Die Felder im Payload sind wie folgt:

Feldname	Type	Beschreibung
`insurantId`	`string`	Die Versichertennummer (KVNR) des Patienten, dessen Akte lokalisiert werden soll
`ignoreCache`	`bool`	Boolean-Flag, ob der Cache ignoriert werden soll (optional, Default: `false`)

Beispiel 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:

ErrorCodes	Bedeutung	Verantwortung / Lösung
`RECORD_EXISTS_AND_ACTIVATED`	KVNR ist bei einem der Aktensystembetreiber registriert und aktiviert	-

`RECORD_NOT_FOUND`	KVNR wurde in keinen der Aktensystembetreiber gefunden.	User: muss sich mit KVNR bei einem Aktensystembetreiber registrieren.
`RECORD_STATUS_CONFLICT`	KVNR wurde zwar bei einem Aktensystembetreiber gefunden, scheint aber nicht aktiviert zu sein (ePA ist deaktiviert)	User: muss sich bei dem angemeldeten Aktensystembetreiber wieder freischalten lassen.