gematik Authenticator SDK Dokumentation

Diese Dokumentation richtet sich an Software-Entwickler, die den Gematik Authenticator für eine Smartcard-basierte Authentisierung an ihre Fachanwendung bzw. ihren Fachdienst anbinden möchten.

Für Rückfragen von Anwendungsentwicklern bzgl. der API bitte die folgende E-Mail-Adresse verwenden: authenticator@gematik.de.

Um Ihnen bei Ihren Fragen oder Störungen am schnellsten weiterhelfen zu können, bitten wir Sie (falls möglich), folgende Informationen in der E-Mail zu hinterlegen:

- Welche Version des Authenticators ist im Einsatz?
- Für welche Anwendung wird der Authenticator genutzt?
- Kurze Problembeschreibung.
- Falls möglich, direkt Screenshots und/oder einen Log-Auszug in der Mail anhängen.

Eine Weitergabe dieser E-Mail-Adresse an die Leistungserbringer ist zunächst nicht vorgesehen.

Bestandteile:

- Authenticator Installer (Installiert auf Windows-Rechnern die Authenticator Executable)
- Authenticator Repository (zum Bauen der Authenticator-Anwendung)
- Source-Code für Beispiel-Aufruf Authenticator aus Anwendung

Inhaltsverzeichnis

Vorbereitung

Was muss ich im Vorfeld beachten/sicherstellen damit der Authenticator erfolgreich funktioniert?

Der Authenticator ist eine Desktop-Anwendung mit grafischer Benutzerschnittstelle, welche aktuell unter Windows lauffähig ist und aus Anwendungen (typisch: Web-Anwendungen) heraus aufgerufen wird.
Seine Aufgabe ist die Authentisierung des Nutzers an einem Identity Provider (IdP) mittels Smartcards der TI (HBA, SMC-B) und Konnektors/KT (2-Faktor-Authentisierungsverfahren). Der Authenticator ist somit eine Anwendung für das Authentifizierungsverfahren und muss daher als Schnittstelle fest in die Fachanwendung eingebunden werden.

Damit der Authenticator für entsprechende Anwendungen betrieben werden kann, ist die Installation der Software sowie die einmalige Administration der Einstellungen durch einen Fachdienst vor Ort oder Administrator des Instituts notwendig.

Was ist notwendig für die Nutzung des Authenticators?

Aus Sicht einer Anwendung:
Damit der Authenticator für eine neue Anwendung auch erfolgreich genutzt werden kann, muss im Vorfeld eine jeweilige Registrierung beim entsprechenden IDP erfolgt sein.
Sie erhalten bei der Registrierung die notwendigen Parameter, die für die Konfiguration des Authenticators sowie die Requests aus der Fachanwendung heraus notwendig sind.

Für eine initiale Registrierung wenden Sie sich bitte an folgende Adresse: IDP-Registrierung@gematik.de.

Aus Sicht eines LE/LEI:
Damit der Authenticator für entsprechende Anwendungen betrieben werden kann, ist die Installation der Software sowie die einmalige Administration der Einstellungen durch einen Dienstleister vor Ort oder Administrator des Instituts notwendig.
Die entsprechenden Einstellungen entnehmen Sie bitte vom Anwendungsverantwortlichen.

Installation, Einrichtung & Nutzung des Authenticators

1) Installation:

Die fachgerechte Installation des Authenticators ist im Installationshandbuch beschrieben.

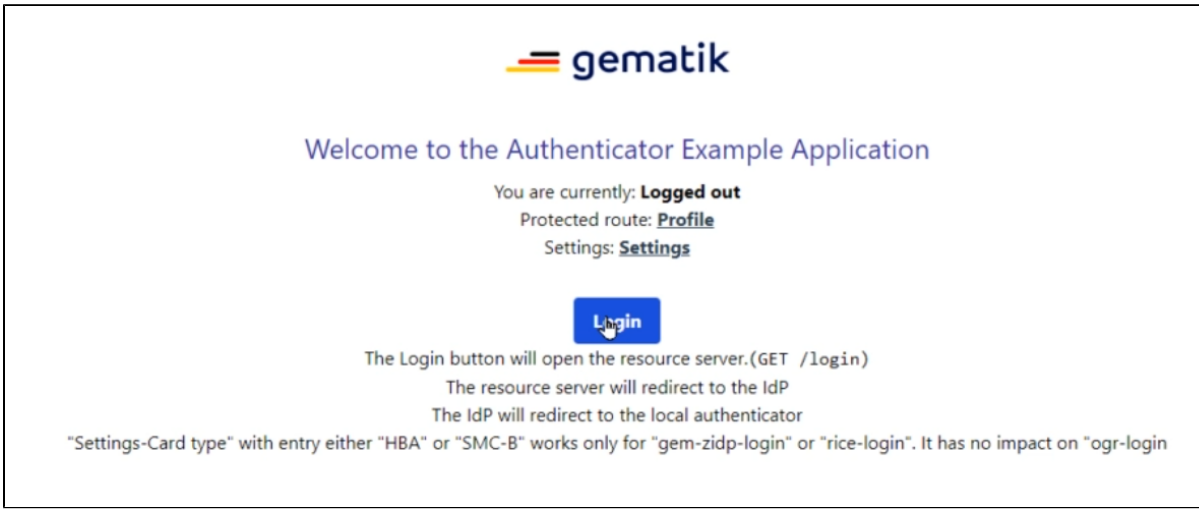
2) Einrichtung:

Die Beschreibung zur Einrichtung finden Sie unter Einstellungen für Verbindung zum Konnektor sowie innerhalb der Schnittstellenbeschreibung.

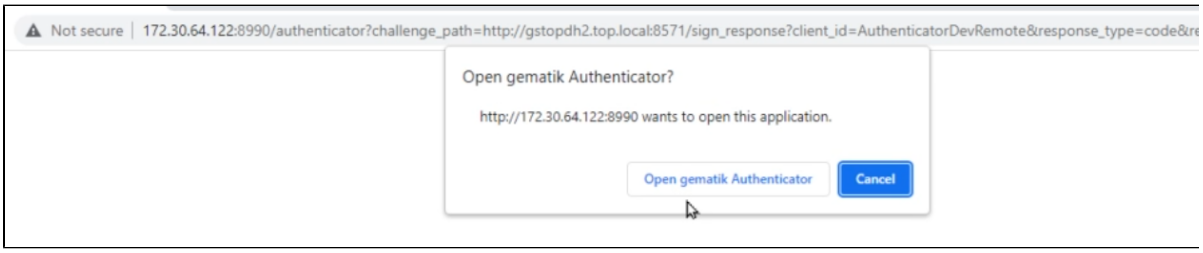
3) Nutzung:

1. Als Nutzer meldet man sich in der Web-Anwendung an und anschließend wird der Authenticator gestartet und führt den Nutzer durch den Anmeldeprozess.
    - Bspw. über Deeplink Absprung der Fachanwendung zum Authenticator

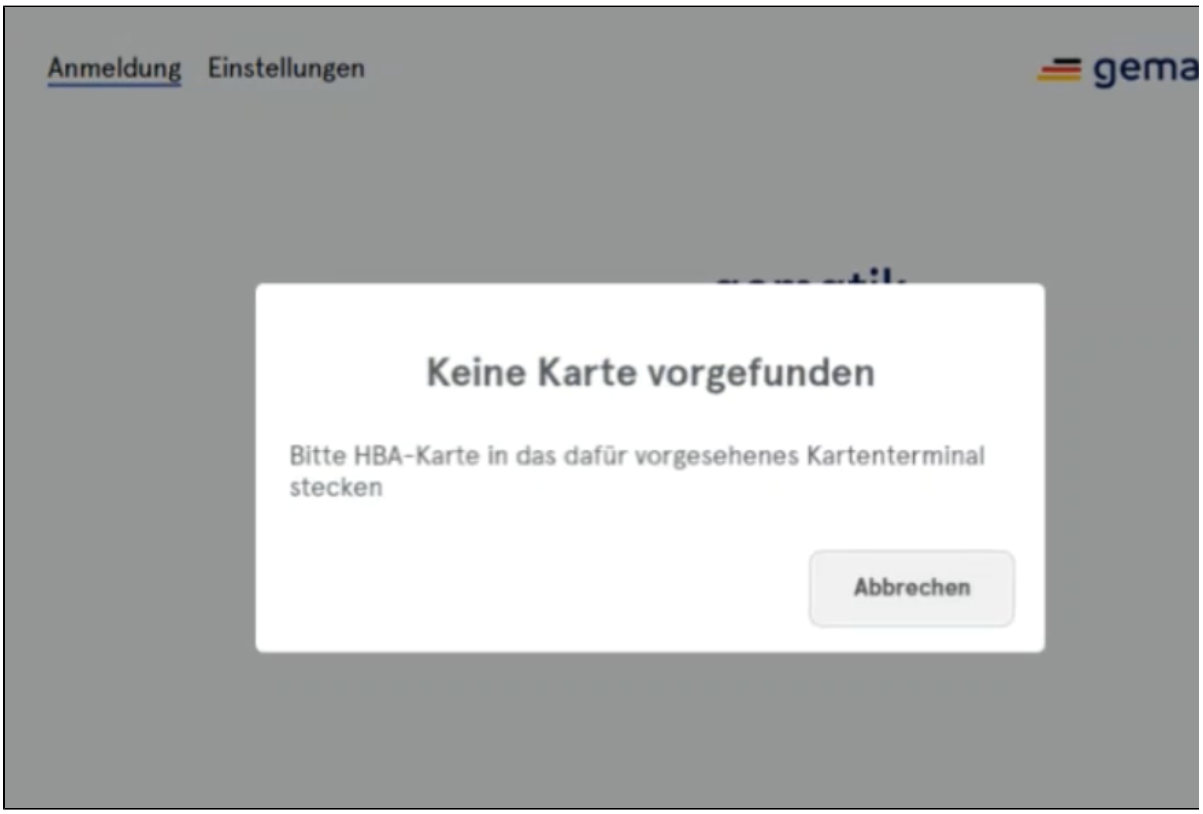
      Anmerkung: Folgende Abbildung zeigt eine Beispielanwendung.



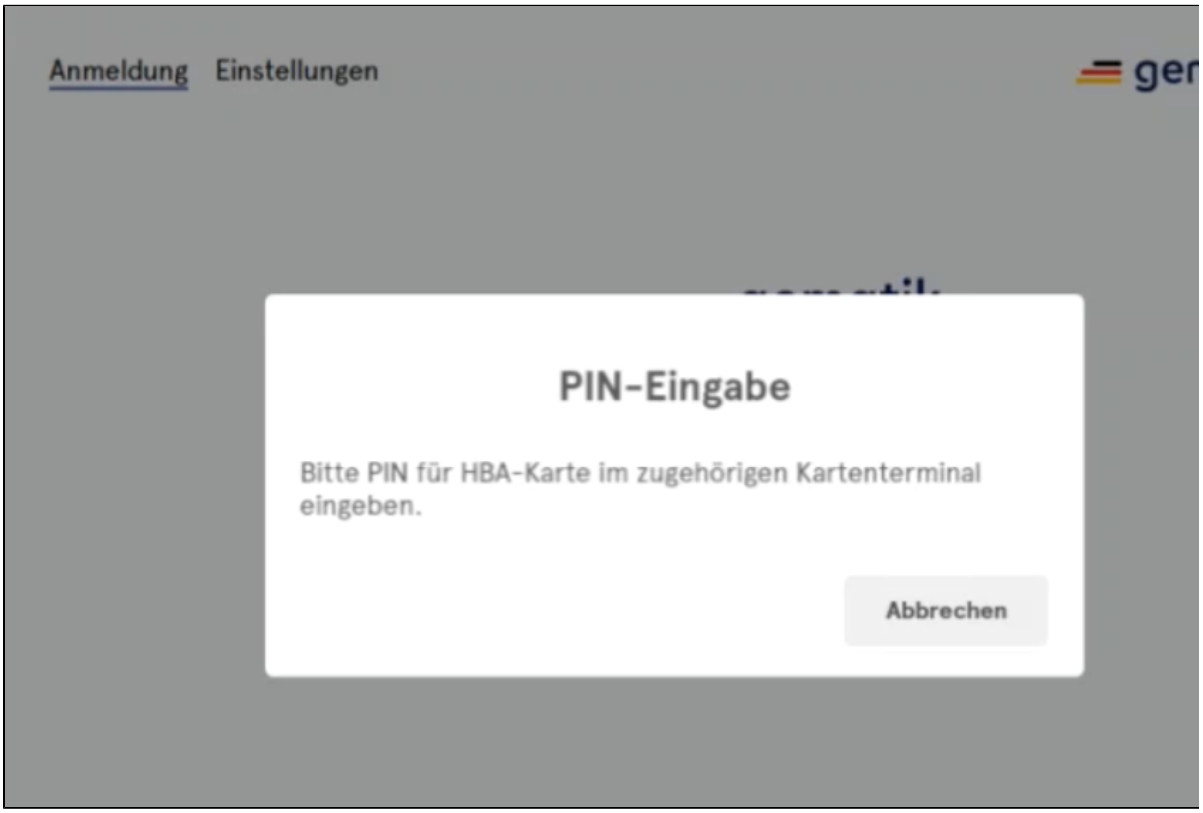
2. Die Anfrage in der Fachanwendung löst den Authentifizierungsprozess aus.
3. Je nachdem, ob die Anmeldung mit HBA oder SMC-B erfolgt, wird hier via Konnektor ein entsprechendes Zertifikat abgefragt.
    - Deeplinkaufruf über die Anwendung:



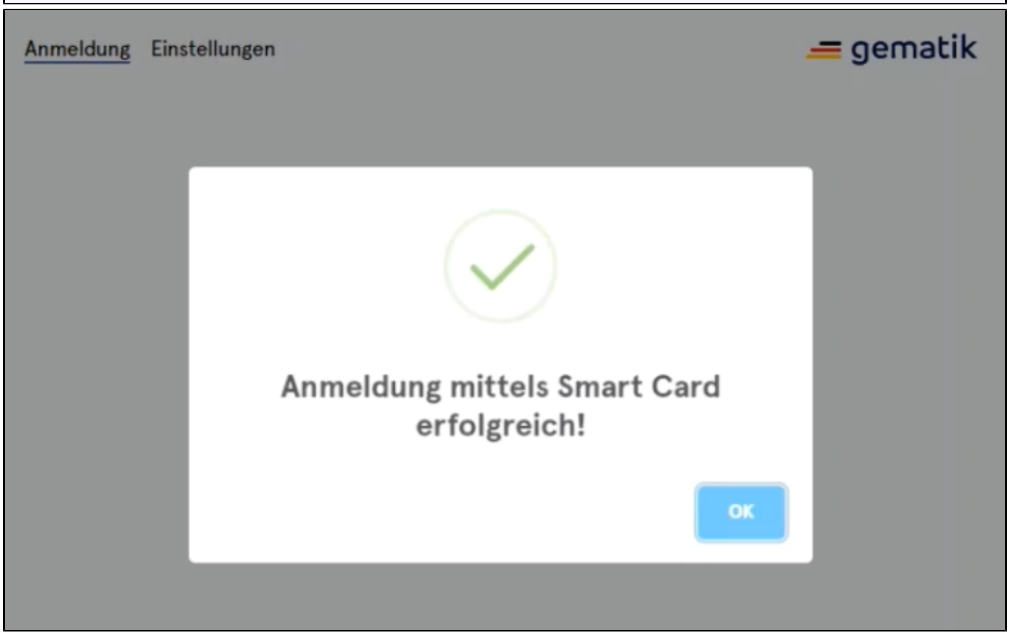
4. Prüfung ob entsprechende Karte gesteckt ist. Falls nicht, wird aufgefordert, die Karte zu stecken.



5. Die Ärztin oder der Arzt müssen nun eine PIN eingeben, die ihnen eindeutig zugeordnet ist.



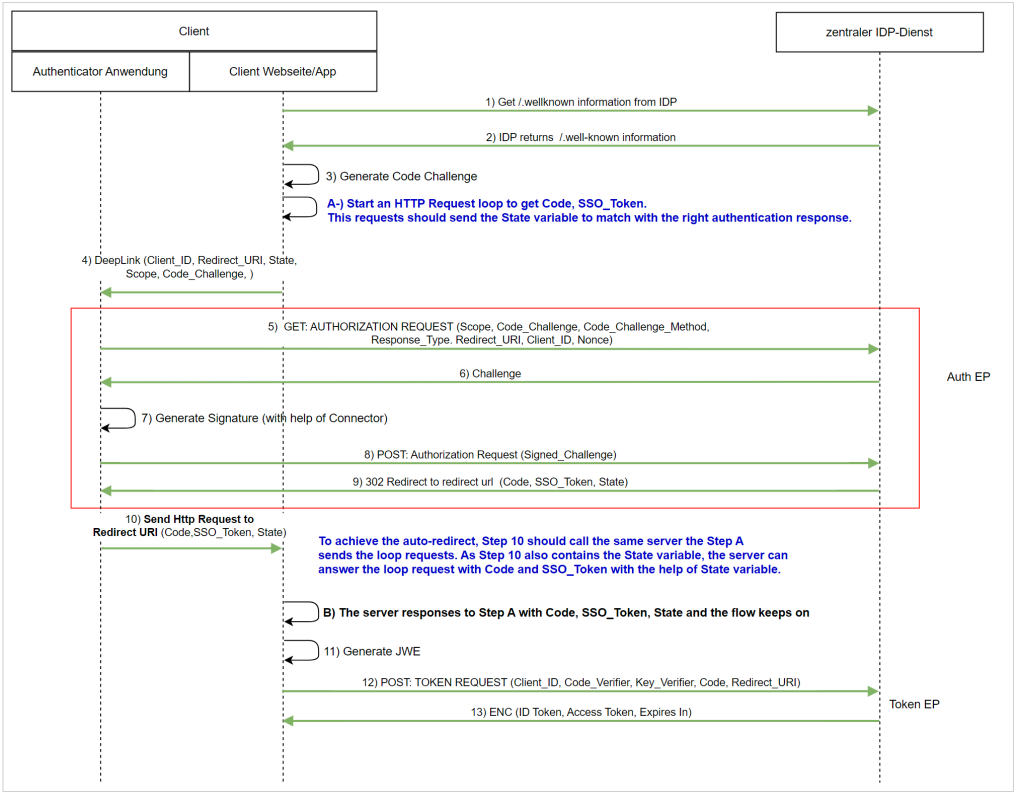
6. Daraufhin erfolgt eine Überprüfung durch einen externen Identitäts-Dienst.



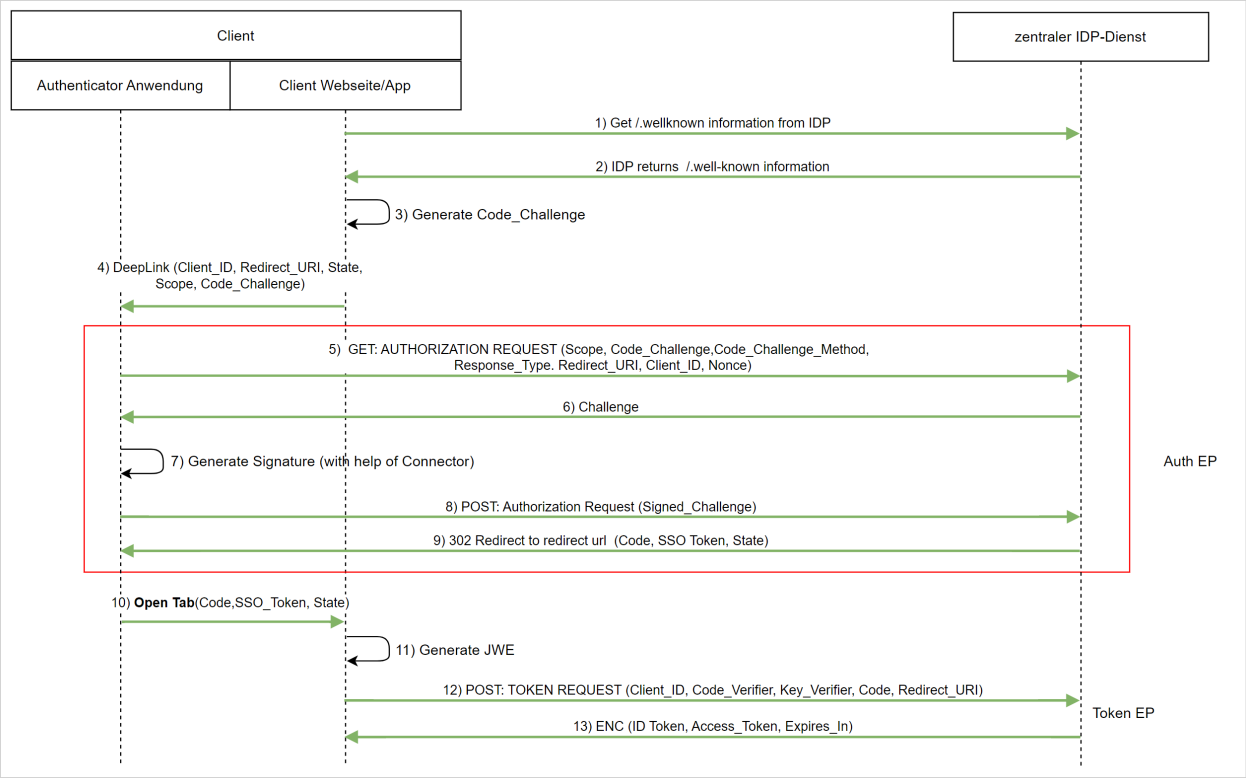
7. Ist die Überprüfung erfolgreich, kehrt der Arzt oder die Ärztin in die Fach-Anwendung zurück und kann darin arbeiten.

Nutzungsszenarium für den Authenticator

Nutzung mit Auto-Redirect



Nutzung mit Deeplink



Einstellungen für Verbindung zum Konnektor

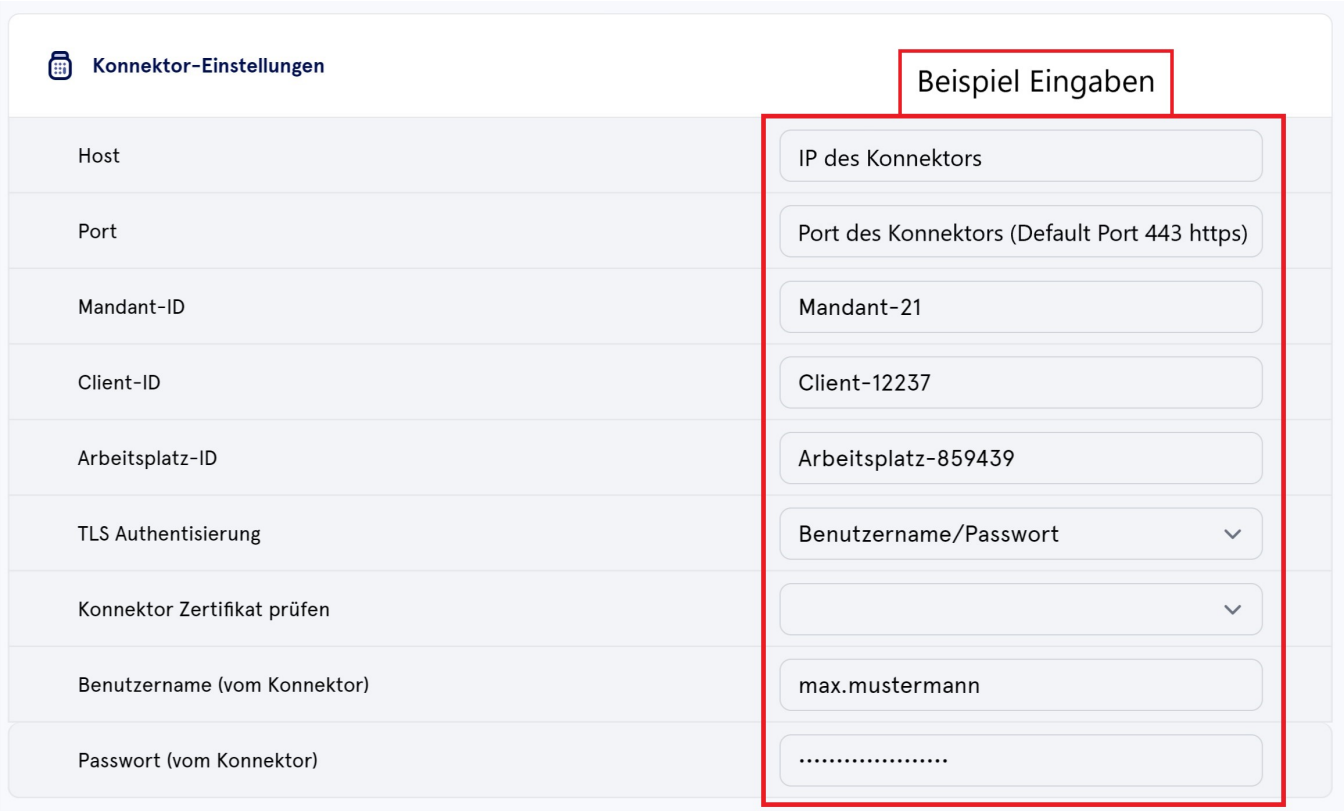
Unter Einstellungen werden die konnektorspezifischen Einstellungen des Authenticators hinterlegt.

Diese werden im Anschluss in einer Konfigurations-Datei unter **..\Users\{$username}\AppData\Local\gematik Authenticator\config.json** abgelegt.

Woher bekomme ich die entsprechenden Konnektor Einstellungen?

Für jeden Konnektor gibt es entsprechende Admin-Handbücher von den jeweiligen Herstellern, in welchen beschrieben sind, was an welcher Stelle konfiguriert werden kann und sollte.

Was am aktuellen Konnektor konfiguriert wurde, weiß i.d.R. der Nutzer/Admin des Konnektors bescheid. Die jeweiligen Werte können in der Konnektor-Admin-Oberfläche aufgefunden werden und müssen dementsprechend beim Admin oder Betreiber des Konnektors angefragt werden.



Konnektoreinstellungen für Produktivbetrieb

Name	Beschreibung	Erforderlich	Format	Key	Beispiel	Default
Host	Hostname / IP vom Konnektor	Ja	URL	connector.entryOption.hostname	127.0.0.1	
Port	Port des Konnektors	Ja	NUMBER	connector.entryOption.port	443 Der Authenticator unterstützt nur gesicherte https Verbindungen.	
Mandant-ID	ID vom Mandant	Ja	LETTERS_NUMBERS	connector.contextParameter.mandantId	Mandant-x	
Client-ID	ID vom Client	Ja	LETTERS_NUMBERS	connector.contextParameter.clientId	Client-System-x	null
Arbeitsplatz-ID	ID vom Arbeitsplatz	Ja	LETTERS_NUMBERS	connector.contextParameter.workplaceId	Arbeitsplatz-x	
TLS mit Authentisierung	Gesicherte Verbindung erforderlich		Enum (BasicAuth; ServerCertAuth; ServerClientCertAuth, ServerClientCertAuth_Pfx)	connector.tlsAuthType	Auswählbare Möglichkeiten: • BasicAuth • ServerCertAuth • ServerClientCertAuth • ServerClientCertAuth_Pfx	ServerCertAuth
Privater Schlüssel	Eine .pem-Datei mit private-key	Ja, falls tlsAuthType 'ServerClientCertAuth' ist	FILE_PATH	connector.entryOption.keyFile	Kann vom Konnektor generiert werden Bsp.: C:/Users/{Nutzerverzeichnis}/AppData/Local/gematik Authenticator/Client-Konnektor-TLS-EE-Key.pem	null

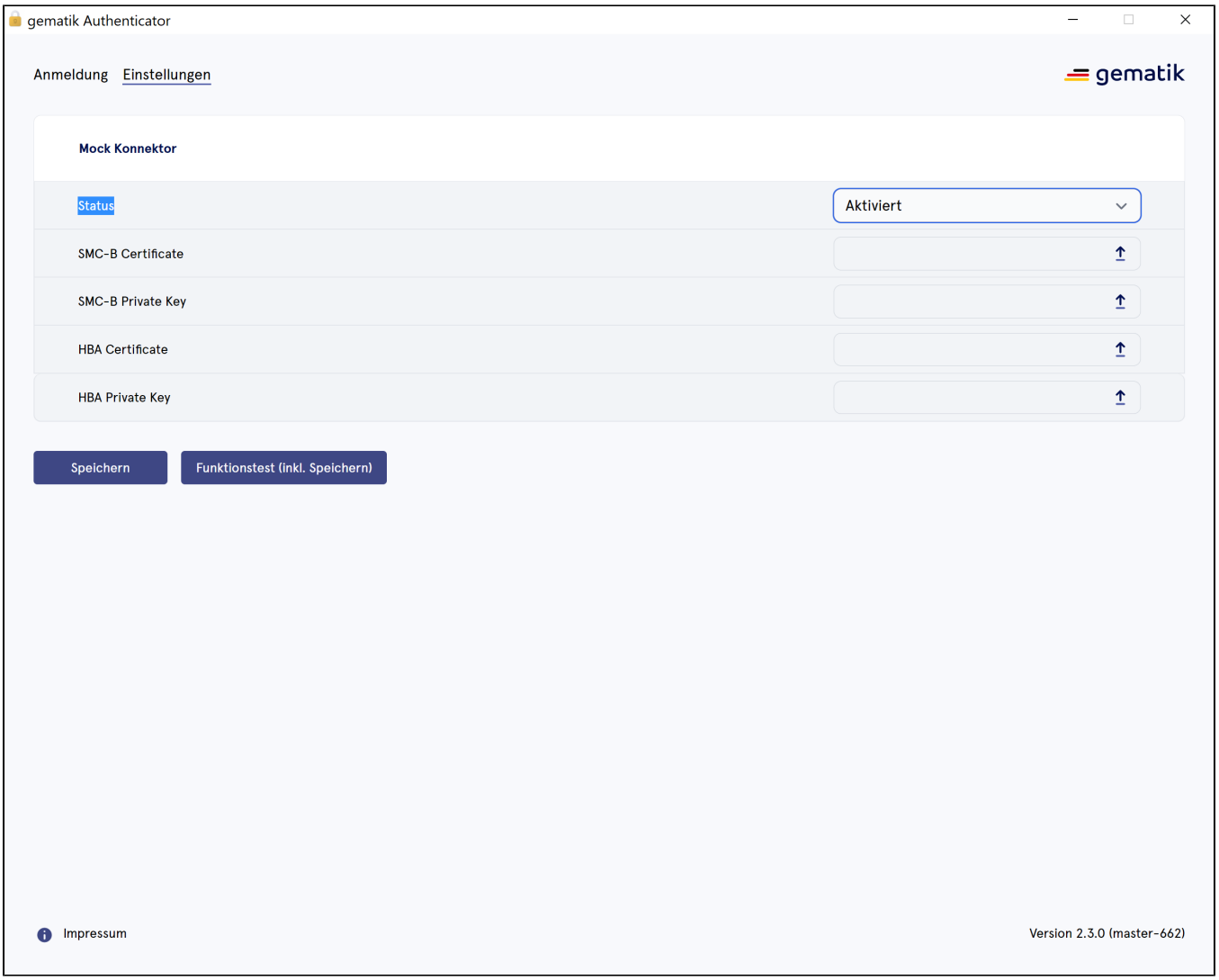
Client-Zertifikat	Eine .pem-Datei mit Client-Zertifikat	Ja, falls tlsAuthType 'ServerClientCertAuth' ist	FILE_PATH	connector.entryOption.certFile	Kann vom Konnektor generiert werden Bsp.: C:/Users/{Nutzerverzeichnis}/AppData/Local/gematik Authenticator/Client-Konnektor-TLS-EE-Cert.pem	null
P12-Datei	Eine p12-Datei, die ein RSA Zertifikat enthält	Ja, falls tlsAuthType 'ServerClientCertAuth_Pfx' ist	FILE_PATH	connector.entryOption.pfxFile	Wird im Konnektor generiert Bsp.: C:/Users/{Nutzerverzeichnis}/AppData/Local/gematik Authenticator/Client-Konnektor-TLS-EE.p12	null
Passwort der P12-Datei (pfx-Datei)	Passwort der p12-Datei (pfx-Datei)	Ja, falls tlsAuthType 'ServerClientCertAuth_Pfx' ist	String	connector.entryOption.pfxPassword	Wird beim Generieren der P12-Datei festgelegt	null
Benutzername	Wird im Konnektor für BasicAuth angelegt	Ja, falls tlsAuthType 'BasicAuth' ist	String	connector.entryOption.username	"user"	''
Passwort	Wird im Konnektor für BasicAuth angelegt	Ja, falls tlsAuthType 'BasicAuth' ist	String	connector.entryOption.password	"password"	''

Zusätzliche Einstellungen für Entwicklervariante

Mockmodus

Innerhalb der Entwicklervariante des Authenticators (***gematik Authenticator Setup - Mock Version X.X.X.exe***) ist ein Mockmodus integriert, der die Verwendung eines Konnektors simulieren kann. Somit können Funktionstests auch ohne physisch vorhandenen Konnektor durchgeführt werden. Diese Funktion soll die Entwicklung mit dem Authenticator vereinfachen, da sie neben einem speziellen Mockmodus auch mehr Logging-Möglichkeiten zur Verfügung stellt.

Hinweis: *Eine detaillierte Dokumentation über den Mockmodus innerhalb der Entwicklervariante ist in unserer Wissensdatenbank hinterlegt: Gematik Authenticator - Entwicklervariante Mockmodus*



Nach der Aktivierung erscheinen vier Möglichkeiten für die Dateieingabe:

Nutzung SMC-B

- SMC-B Zertifikat
- SMC-B privater Schlüssel

Nutzung HBA

- HBA Zertifikat
- HBA privater Schlüssel

Wählen Sie die zugehörigen Zertifikate (*im .pem- oder .p12-Format*) aus, um diese Funktion ordnungsgemäß zu verwenden. Beispielzertifikate werden vom Authenticator Team der Gematik GmbH bereitgestellt - Sie können jedoch jederzeit Ihre Zertifikate und privaten Schlüssel verwenden, um die Mockmodus-Funktion zu nutzen.

Hinweis: Sollten Ihnen die benötigten Zertifikate bzw. Schlüssel im p12-Format vorliegen, können Sie diese in das benötigte .pem-Format exportieren. Eine detaillierte Anleitung, für einen möglichen Weg, finden Sie in unserer Wissensdatenbank.

Logging

Innerhalb der Entwicklervariante wird der ganze Netzwerk-Traffic des Authenticators (Requests & Responses) innerhalb des Logfiles zentral gepflegt.

Pfad zum Logfile
`C:/Users/{Nutzerverzeichnis}/AppData/Local/Temp/authenticator-logging/authenticator-${datum}.log`

Einstellungen für Proxy

Da es in Krankenhäusern sehr üblich ist, einen Proxy zu verwenden, unterstützen wir auch die Proxy-Funktion. Der Authenticator unterstützt Basic Auth und Client Certificate Auth für den Proxyserver. Um die Proxy-Funktion zu verwenden, aktivieren Sie sie im Abschnitt „Proxy“ auf der Seite „Einstellungen“.

Feldname	Beschreibung	Beispiel
Proxy-Authentifizierung	Proxy Typ	Basic Authentifizierung oder Zertifikats-Authentifizierung
Benutzername	Benutzername (Basic Authentifizierung)	p_user
Passwort	Passwort (Basic Authentifizierung)	p_password
Client-Zertifikat	Passwort (Zertifikats-Authentifizierung)	PEM Zertifikat für Proxy Server

---

Schnittstellenbeschreibung (API)

NEW Auto-Redirect Funktion (ab v4.0.0)

Bisher hat der gematik Authenticator die Parameter in der Form „authenticator://params“ von einer Webseite empfangen und nach Abschluss des Vorgangs das Ergebnis in dem default Browser des Betriebssystems geöffnet.

Um das Öffnen in dem default Browser zu verhindern, wurde die Auto-Redirect Funktion entwickelt. Mit dieser Funktion verarbeitet der Authenticator einen zusätzlichen Parameter: "callback=DIRECT". Durch diesen ruft der Authenticator die Redirect-URI direkt auf, anstatt das Ergebnis der Authentifizierung in einen neuen Browser-Tab zu öffnen.

Der Empfänger der Redirect-URI muss dazu einen GET-Endpunkt bereitstellen, welcher der Redirect-Uri entspricht. Die weitere Verarbeitung obliegt dem Empfänger.

Implementierung der Auto-Redirect Funktion für Entwickler:

Verwende die “callback” Parameter in der Authenticator-App:
Der Authenticator muss mit dem Parameter callback=DIRECT aufgerufen werden, damit die Auto-Redirect Funktion aktiviert und die Redirect-URI direkt aufgerufen wird.

Erstelle einen GET-Endpoint für die Redirect-URI:
Der Server muss einen REST GET-Endpoint zur Verfügung stellen, welcher der Redirect-URI entspricht. Dieser wird vom Authenticator aufgerufen wenn der Vorgang abgeschlossen ist.

Erstelle eine sichere Verbindung zwischen dem OIDC-Client und dem Server:
Um eine sichere Verbindung zwischen dem OIDC-Client und dem Server herzustellen, kann eine Verifikationszeichenfolge (verifier string) über PKCE (Proof Key for Code Exchange) zur Authentifizierung verwendet werden.

Während der Authentifizierung ist ein Fehler aufgetreten

Wenn ein Fehler während des Authentifizierungsvorgangs auftritt, ruft der Authenticator die Redirect-URI auf. Die Fehlermeldung ist OAuth2 konform und erhält als zusätzlichen Parameter den STATE um eine Zuordnung zu ermöglichen.

Deeplink mit Beispielaufruf zentraler IDP (RU)

Die Fachanwendung kann den Authenticator über einen "Deeplink" nach Rückfrage an den Anwender direkt aus dem Browser starten.
Beim Deeplink-Aufruf übergibt die Fachanwendung einen URL-String mit Query-Parametern. Dieser URL-String setzt sich abhängig vom verwendeten IDP aus dem Protokoll ("authenticator://") und abhängig vom verwendeten IDP genannten Parametern für den Login-Endpoint zusammen. (Die Teilparameter können auch base64 kodiert sein.

Hinweis: *Bitte beachten: Sonderzeichen wie ':' und '/' sind dabei zu encoden (hier '.'=> '%3a' und '/' => '%2F'))*

Bei diesem Aufruf wird immer der Default-Browser vom Betriebssystem verwendet. Zudem erfolgt der Aufruf in einem neuen Tab.

Beispiel Deeplink zentraler IDP:

```
"authenticator://?challenge_path=https://idp-ref.app.ti-dienste.de/auth?
client_id=GEMgematAut5zGBeGaqR&response_type=code&redirect_uri=https%3A%2F%2Fgstopdh4.top.local%3A8090%
2Fcallback&state=f1bQrZ4SEsiKCRV4VNqG&code_challenge=JvcJb54WkEm38N3U1IYQsP2Lqvv4Nx23D2mU7QePWEw&code_challen
ge_method=S256&scope=openid+gem-auth&nonce=MbwsuHIExDKyqKDKSsPp&cardType=SMC-B"
```

wobei der String sich aus folgenden mittels "?" konkatenierten Teilen zusammensetzen kann:

Key	Beschreibung	Beispiel
"authenticator://"	Protokoll für Authenticator-Deeplink. Immer "authenticator://" **HKEY_CURRENT_USER\Software\Classes\authenticator**	"authenticator://"
"challenge_path"	Der Challenge/IDP Path setzt sich aus dem **authorization_endpoint** und den u.g. Parametern zusammen. Den konkreten **authorization_endpoint** müssen Sie sich aus dem Discovery Document des IDPs holen! Bitte beachten: Sonderzeichen wie ':' und '/' sind dabei zu encoden (hier '.'=> '%3a' und '/' => '%2F').	"https://idp-ref.app.ti-dienste.de/auth? client_id=GEMgematAut5zGBeGaqR&response_type=code&redirect_uri=https%3A% 2F%2Fgstopdh4.top.local%3A8090% 2Fcallback&state=f1bQrZ4SEsiKCRV4VNqG&code_challenge=JvcJb54WkEm38N3U1IY QsP2Lqvv4Nx23D2mU7QePWEw&code_challenge_method=S256&scope=openid+gem- auth&nonce=MbwsuHIExDKyqKDKSsPp&cardType=SMC-B"

Der Challenge Path setzt sich aus folgenden Parametern zusammen:

auhorization_endpoint	Innerhalb dieses Discovery Documents ist der authorization_endpoint auffindbar. Der authorization_endpoint wird zusammen mit den u.g. Parametern verwendet, um die challenge_path zu bilden. Zentraler IDP RU: https://idp-ref.app.ti-dienste.de/.well-known/openid-configuration Zentraler IDP PU: https://idp.app.ti-dienste.de/.well-known/openid-configuration	"https://idp-ref.app.ti-dienste.de/auth"
"client_id"	Die client_id des Clients. Wird bei Registrierung beim IDP vergeben.	"GEMgematAut5zGBeGaqR"
"response_type"	Referenziert den erwarteten Response-Type des Flows. Muss immer 'code' lauten. Damit wird angezeigt, dass es sich hierbei um einen Authorization Code Flow handelt. Für eine nähere Erläuterung siehe OpenID-Spezifikation.	"code"

"redirect_uri"	Die für den Client beim Server hinterlegte redirect_uri. Muss dem bei der Registrierung hinterlegten Wert entsprechen. Bitte beachten: Sonderzeichen wie ':' und '/' sind dabei zu encoden (hier '.'=> '%3a' und '/' => '%2F')	"https%3A%2F%2Fgstopdh4.top.local%3A8090%2Fcallback"
"state"	Der state der Session. Sollte dem zufällig generierten state-Wert aus der initialen Anfrage entsprechen. **Hinweis**: Darf nicht länger als 32 Zeichen sein (PU). Innerhalb der RU wurde der zugelassene Wert auf max. 512 Zeichen erhöht.	"f1bQrZ4SEsiKCRV4VNqG"
"code_challenge"	Der Hashwert des Code-Verifiers wird zum IDP als Code-Challenge gesendet.	"JvcJb54WkEm38N3U1IYQsP2Lqvv4Nx23D2mU7QePWEw"
"code_challenge_method"	Das Primärsystem generiert einen Code-Verifier und erzeugt darüber einen Hash im Verfahren SHA-256. Bei dem zentralen IDP ist es immer S256.	"S256"
"scope"	Der Scope entspricht openid plus dem zwischen Fachdienst und IDP festgelegten Wert. Der Scope besteht grundsätzlich aus zwei Parametern: 1. Open ID 2. gem-auth Eigener registrierter Scope im zentral IDP **Hinweis:** Der Parameter "gem-auth" dient nur zum Testen des gematik-Authentifikators. Daher kann dieser nicht innerhalb der Produktion genutzt werden. Um einen eigenen Scope für ihre Fachanwendung zu erhalten, müssen Sie zuvor eine Fachdienstregistrierung (siehe FAQ) beim IDP durchführen.	"openid gem-auth"
"nonce"	String zur Verhinderung von CSRF-Attacken. Dieser Wert ist optional. Wenn er mitgegeben wird muss der gleiche Wert im abschließend ausgegebenen ID-Token wieder auftauchen. **Hinweis**: Darf nicht länger als 32 Zeichen sein.	"MbwsuHIExDKyqKDKSsPp"
"callback"	Optionaler Parameter, siehe "Auto-Redirect Funktion (ab v4.0.0)"	"DIRECT"
"cardType"	Der Kartentypparameter kann HBA, SMC-B oder multi sein. Der Parameter "multi" entspricht der Kombination aus HBA und SMC-B	HBA | SMC-B | multi

Beachte: Bei einem Fehler des Deeplink-Login-Aufrufs wird der Fehler "login_not_successful: 'Anmeldung mittels Smart Card nicht erfolgreich!'" zurück geliefert.

Die im Verlauf auftretenden IDP/Flow Meldungen werden an die aufrufende Anwendung weitergereicht und sind in folgender Fehrcode-Liste enthalten: Authenticator Fehlercodes

---

Fehlercodes

Alle aktuellen Fehlercodes können Sie auf folgender Seite einsehen: Authenticator Fehlercodes

---

Bauen der Anwendung

Schritt 1: Anwendung erstellen

- Projekt Klonen
- Gewünschten Branch auschecken
- Mit „npm install“ alle nötigen Dependencies installieren

Schritt 2: Anwendung starten

- „npm run dev“

Noch Fragen?

Vielleicht finden Sie hier eine Antwort: Fragen & Antworten (FAQs)

Weiterführende Links

https://gematik.github.io/ref-idp-server/tokenFlowEgk.html#e8a4b1ed-138e-4758-b3dd-cc2c900696ab

Installationshandbuch Authenticator

https://fachportal.gematik.de/
