### INFORMATIONEN ZUR NUTZUNG DER ID-ISSUER API

**Bundesdruckerei GmbH** Kommandantenstraße 18 10969 Berlin



Zur Nutzung der API der ID-Issuer Test-Instanz müssen Sie sich nach Open-ID Connect (OIDC) authentisieren. Des Weiteren benötigen Sie ein TLS-Client-Zertifikat. Das Zertifikat erhalten Sie nach erfolgter Registrierung via E-Mail, sowie das dazugehörige Passwort via SMS-Nachricht.

Zum Abholen des OIDC-Access-Tokens muss folgende URL angesprochen werden:

#### Public-Testsystem:

https://iam.test.de.id-issuer.eu/auth/realms/idissuer/protocol/openid-connect/token

#### Produktivsystem:

https://iam.de.id-issuer.eu/auth/realms/idissuer/protocol/openid-connect/token

Der POST-Request zur Anfrage eines Access-Token muss folgende Parameter enthalten:

Parameter	Wert
clientId	Fester Wert "self-management-gui"
clientUsername	Erhält man über des Selfmanagementportal
clientPassword	Erhält man über des Selfmanagementportal
clientSecret	Erhält man über des Selfmanagementportal
grantType	Fester Wert "password"

#### Beispiel zum Abholen des Access-Token mit CURL:

#### Initial:

curl -d "grant\_type=password&username=[Erhält man im SP]&password=[Erhält man im SP]&client\_id=self-management-gui&client\_secret=[Erhält man im SP]" -H "Content-Type: application/x-www-form-urlencoded" -X POST "https://keycloak.test.de.idissuer.eu/auth/realms/idissuer/protocol/openid-connect/token"

#### Mit Werten belegt:

curl -d "grant\_type=password&username=Uhhkzzh6&password=Jgfou75!&client\_id=selfmanagement-gui&client\_secret=3d308e29-13d6-46eb-be3b-567ac5275277" -H "Content-Type: application/x-www-form-urlencoded" -X POST "https://keycloak.test.de.idissuer.eu/auth/realms/idissuer/protocol/openid-connect/token"

## <sup>6</sup>bundesdruckerei.

#### Nutzung der API mit Postman

Für die Nutzung der API mit Postman sind zwei Dinge notwendig: Zum einen muss das TLS-Client-Zertifikat eingebunden werden und zum anderen muss der Access-Token abgeholt werden. Des Weiteren kann man die YAML-Datei direkt als Collection in Postman importieren.

#### 1. TLS-Client-Zertifikat in Postman einbinden.

Das Tool Postman kann mit P12-Container nicht umgehen. Daher muss der private Schlüssel als PKCS8 und die Zertifikatskette als PEM-Format (base64) aus dem P12-Container exportiert werden. Anschließend kann man unter den Einstellungen von Postman das TLS-Client-Zertifikat einbinden, wie in der nachfolgenden Darstellung abgebildet.

General The Add Client Cert	ificate	Shortcuts	Data	Add-ons	Certificate	es Prox	y Update	About
Host	https://	*.de.id-issi	uer.eu		:	443		
CRT file	C:\Users\pam-admin\Documents\id_issuer_test.de.cer Choose					Choose File		
KEY file	C:\Users\pam-admin\Documents\id_issuer_test.de.key Choose Fil				Choose File			
Passphrase	123456							
							Cancel	Add

# <sup>6</sup>bundesdruckerei.

### 2. YAML-Datei importieren

Die YAML-Datei kann importiert werden und anschließend erhält man eine Collection, welche alle Requests für die ID-Issuer-Instanz aufführt.

🥝 Postman		
File Edit View Help		
🕂 New 🔻 Import Runner 🖡 🔻	My Works	space 🔻 🌲 Invite
Q Filter	GET Untitled Request X + ····	
History Collections	Untitled Request	
Trash Cl	GFT T Enter request URL	
<ul> <li>Tobacco Track&amp;Trace ID Issuer API 15 requests</li> </ul>	Params Authorization Headers Body Pr	e-request Script Tesi
🕨 🖿 order	Query Params	
🕨 🖿 operator	KEY	VALUE
Facility	Key	Value
machine     GET Provides OpenAPI Specification on t	Response	
POST Submit a new order for unit packet		
POST Submit a new order for aggregated		
	Hit	the Send button



#### **3.** Access-Token für die Collection bereitstellen

Um die Requests nutzen zu können, muss noch die Autorisierung für die Collection eingestellt werden. Hierfür klickt man mit der rechten Maustaste auf die Collection und wählt im Kontextmenü "edit" aus. Anschließend klickt man auf den Reiter Authorization und stellt den Autorisierungstyp "OAuth2.0" ein.

EDIT COLLECTION			×	
Name				
Tobacco Track&Trace ID Issuer API				
Description Authorization • Pre-request Script	rs Tests Variable	es 🜒		
This authorization method will be used for every request in	this collection. You can	override this by specifyin	ng one in the request.	
ТҮРЕ	Heads up! Thes	e parameters hold sensi	tive data. To keep this 🛛 🗙	
OAuth 2.0 👻	data secure while working in a collaborative environment, we recommend using variables. Learn more about variables			
The authorization data will be automatically generated when you send the request. Learn more about authorization	Access Token	Access Token Available Tokens 🔻		
Add auth data to		Get New Access To	oken	
Request Headers 👻		L.		
		5	ancel	
		C.	opuate	

Auf die Schaltfläche "Get New Access Token" klicken und die Parameter für den Abruf des Access-Token eingeben. Die Parameter wurden am Anfang des Dokuments beschrieben. Des Weiteren ein Beispiel in der nachfolgenden Abbildung.

Token Name	OIDC Token		
Grant Type	Password Credentials		
Access Token URL 🚺	https://keycloak.test.de.id-issuer.eu/auth/realms/idissuer/protocol/		
Username	Uhhkzzh6		
Password			
	Show Password		
Client ID 🚺	self-management-gui		
Client Secret 🕕	3d308e29-13d6-46eb-be3b-567ac5275277		
Scope 🚺	e.g. readtorg		
Client Authentication	Send as Basic Auth header		



#### 4. Beispiel 3rd-Party-Registration durchführen

Um einen Request auszuführen, ruft man den gewünschten Request aus der Collection auf. Wie zum Beispiel die 3rd-Party-Registration. Zum Ausführen des Requests muss unter dem Reiter Authorization der abgerufene Access-Token eingestellt und im Reiter Body der Request mit Werten gefüllt werden.

Q Filter	POST Submit an order for a 3rd party  + •••				
History Collections	<ul> <li>Submit an order for a 3rd party economic operator registration. Multiple EOs may be</li> </ul>				
Trash	POST v ({baseUrl})/operator				
<ul> <li>Tobacco Track&amp;Trace ID Issuer API</li> <li>15 requests</li> </ul>	Params Authorization Headers (2) Body Pre-request Script Tests				
Morder:     Morder::     Morder:::     Morder::::     Morder:::     Morder:::     Morder::::     Morder::::     Morder:::::::     Morder:::::::::::::::::::::::::::::::::::	none     form-data     x-www-form-urlencoded     rew     binary     jSON (application     )				
	<pre>2. { 3     "position": "1", 4     "details": { 5          "delame1": "Edeka Kreuzberg", 6          "eolame1": "Si", 7          "street": "Oranienstraße", 8          "houseHumber": "Si", 9          "rigCade": 10056", 10          "city": "Berlin" 11          }, 12          "eoCountryReg": "DE", 13          "vetR": "true", 14          "eoExiseHumber1": "true", 15          "eoExiseHumber1": "true", 16          "eoExiseHumber1": "true", 17          "eoExiseHumber1": "true", 18          "eoExiseHumber1": "true", 19          "eoExiseHumber1": "true", 14          "eoExiseHumber1": "true", 14          "eoExiseHumber1": "true", 15          "eoExiseHumber1": "true", 16          "eoExiseHumber1": "true", 17          "eoExiseHumber1": "true", 18          "eoExiseHumber1": "true", 19          "eoExiseHumber1": "true", 19          "eoExiseHumber1": "true", 10          "eoExiseHumber1": "true", 11          "eoExiseHumber1": "true", 12          "eoExiseHumber1": "true", 13          "eoExiseHumber1": "true", 14          "true", 14          "true", 14          "true", 14          "true", 14           "true", 14          "true", 14           "true", 14          "true", 14           "true", 14           "true", 14           "true", 14           "true", 14           "true", 14           "true", 14           "true", 14           "true", 14           "true", 14           "true", 14           "true", 14           "true", 14           "true", 14           "true", 14           "true", 14            "true", 14              "true", 14           "true</pre>				
	Body Cookies Headers (13) Test Results Status: 200				
	<pre>1 * { 2 * {     "idlist": [ 3 * 4</pre>				