EBSW - ErvBox Service Wrapper Beschreibung

Bundesministerium
Justiz

bmj.gv.at

Beschreibung der Schnittstelle zum Upload und Download von Dateien in die ERV-Box

Dateiname: ErvBox_Service_Wrapper_Beschreibung.pdf
Version: 1.0 vom 05.08.2022
Ersteller: Jodi Badr

1. Dokumentinformation

1.1. Inhaltsverzeichnis

1.2. Änderungsverlauf

Version	Datum	Ersteller	Kommentar
1.0	05.08.2022	Jodi Badr	Initiale Version

2. Einleitung

Das ErvBox Wrapper Service wurde als Convenience Schnittstelle für den Zugang zur ERV-Box erstellt. Es soll die Notwendigkeit von zusätzlichen Zertifikaten vermeiden und eine stabile Schnittstelle in der Verantwortung der Servicekette der Justiz bereitstellen. Soweit es möglich war, wurden bereits bekannte und etablierte Sicherheitsmechanismen eingeführt und die Schnittstellen flexibel gestaltet, um eine Erweiterung in Zukunft zu erleichtern.

2.1. Zweck des Dokuments

Dieses Dokument soll die Schnittstelle zum Upload und Download von Dateien mittels ERV-Box beschreiben. Nachfolgend ist der Upload- und Downloadvorgang illustriert.

Die Basis der Verarbeitung ist ein Service Token, welches vom Service Provider über den bereits etablierten Portalverbundzugang erstellt werden kann. Dieses Token wird für jeden Aufruf der weiteren Services benötigt und ist nur einmalig verwendbar. Bei der ersten Verwendung wird das benutzte Token invalidiert und es muss ein neues erstellt werden. Außerdem sind Tokens nur für eine begrenzte Zeit gültig. Die genaue Dauer der Gültigkeit wird in der Beschreibung der Einstellungen der Servicekette hinterlegt.

2.1.1. Anforderung eines Tokens

Vorgehensweise bei der Anforderung eines Tokens:

1. Anfordern eines Tokens.
2. Token wird geliefert.

Beschreibung des Services:

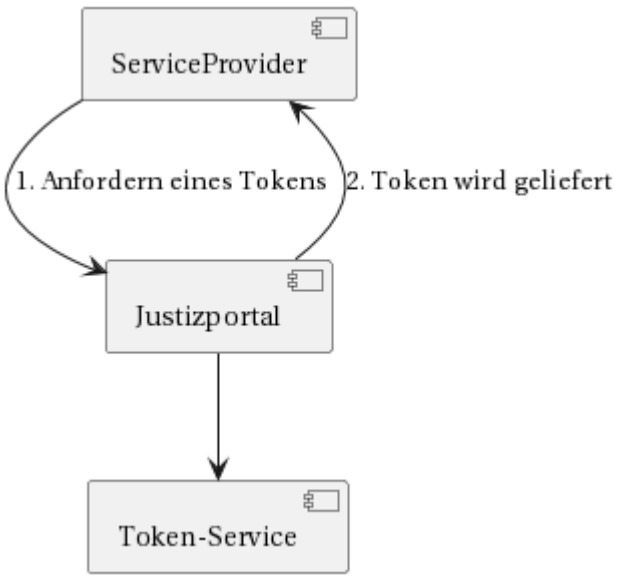
Der Token Service liefert nur ein Token per Aufruf und ist nur für eine Operation gültig, unabhängig ob es sich um einen Upload von Dateien oder Download von Dateien handelt.

Falls die Verbindung bei einem Upload oder Download Aufruf abgebricht, muss ein neues Token angefordert werden und der Vorgang wiederholt werden.

Die Response, falls erfolgreich, beinhaltet den Token selbst und das Ablaufdatum deselben.

Das Ablaufdatum

- bezieht sich auf den spätesten Veerwendungszeitpunkt des Tokens
- *validUntil* wird als Format ISO 8601 (yyyy-MM-dd'T'HH:mm:ss.SSSX) angegeben, Bsp.: 2022-03-22T15:32:00.000+00:00, in diesem Fall handelt es sich um die **UTC-Zeit**.
- ermittelt sich wie folgt: Aktuelle Zeit + 5min, das Ergebnis wird bis zur nächsten vollen Minute aufgerundet
    - Beispiel:
      Aktuelle Zeit= 14:32:12
      Ablaufdatum = 14:32:12 + 5min = 14:37:12 (Minute aufrunden) =14:38:00



2.1.2. Upload von Dateien

Die Vorgehensweise beim Upload:

1. Anfordern eines Tokens
2. Upload mittels Token durchführen

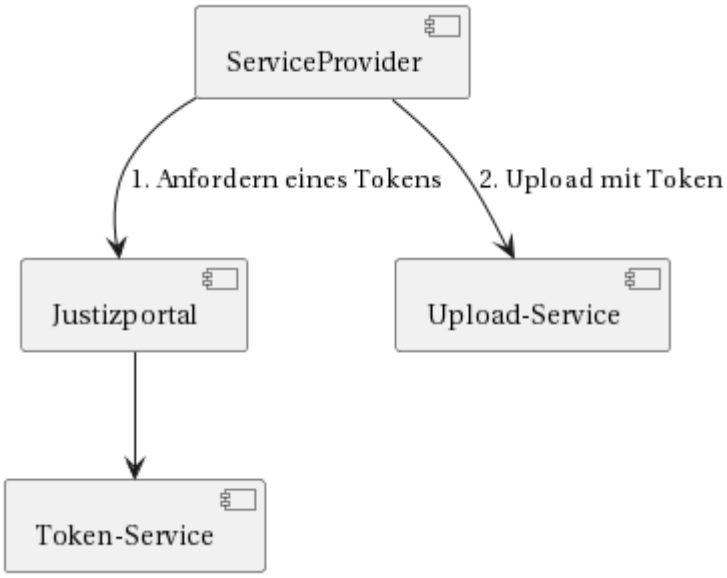
Beschreibung des Services:

Upload von Dateien in die ERV-Box, es ist möglich mehrere Dateien per Request hochzuladen.

Jeder Aufruf muss die Dateien und deren Metadaten enthalten, die Metaddaten in Form von JSON-Struktur.

Die Response falls erfolgreich beinhaltet die Transfer-ID.

Falls die Verbindung bei einem Upload Aufruf abgebrochen ist muss ein neues Token angefordert werden und der Vorgang wiederholt werden.



2.1.3. Download von Dateien

Vorgehensweise beim Download:

1. Anfordern eines Tokens
2. Download der Metadaten eines Transfers mittels Transfer-ID
3. Anfordern eines Tokens
4. Download der einzelnen Dateien mittels File-ID (falls mehrere Dateien runtergeladen werden, muss für jeden einzelnen Datei-Download ein neuer Token verwendet werden, d.h. Wiederholung von Schritt 3 und 4)

Beschreibung des Services:

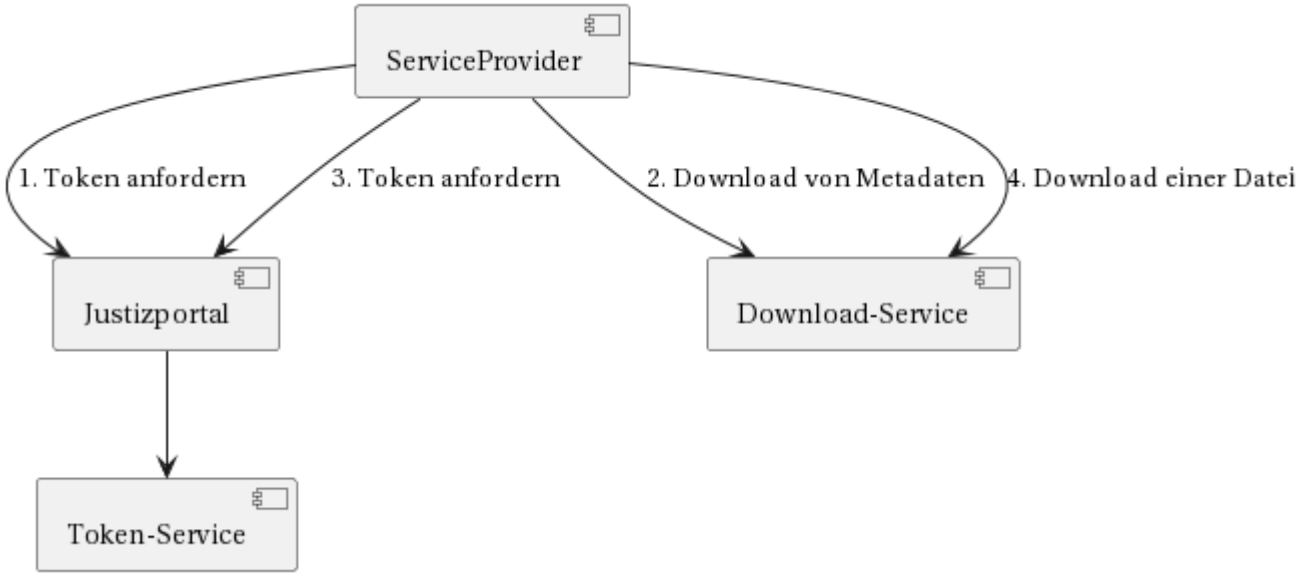
Der Download der Metadaten liefert eine Liste an File-IDs mit rudimentären Metadaten.

Der Download von Dateien selbst hat folgende Einschränkungen:

- es ist ausschliesslich möglich eine Datei per Request herunterladen.
- Im Response wird eine Zip Datei übermittelt, welche zwei Dateien enthält:
    - eine Metadaten Datei
    - die Datei selbst.

Bevor eine Datei runtergeladen werden kann, muss zuerst über die Metadaten-Abfrage die *fileId* der Datei ermittelt werden. Diese Abfrage liefert alle Datei-Metadaten zu einem Transfer.

Falls die Verbindung bei einem Download Aufruf abgebrochen ist muss ein neues Token angefordert werden und der Vorgang wiederholt werden.



3. HTTP-REST-Schnittstelle

3.1. Operationen

Nachfolgend wird jede Operation in einer eigenen Tabelle mit den jeweiligen Request-Parametern und den möglichen Rückmeldungen beschrieben. Bei Fehlerfällen (Response-Codes: 4xx, 5xx) wird für den Transport von Fehlerinformationen an den Aufrufer die nachfolgende JSON-Struktur verwendet:

Response im Fall von Fehlerfällen

```
{
  "errorMessage": <String>
}
```

Der Fehlercode wird mittels HTTP-Statuscode (5xx, 4xx) übertragen, daher wurde in der JSON-Struktur ein entsprechendes Feld nicht vorgesehen.

3.1.1. Token anfordern

Diese Operation retourniert einen neuen Token mit dem ein Upload oder Download durchgeführt werden kann. Die Anforderung eines Tokens erfolgt ohne Request-Parameter. Bevor ein Upload oder Download durchgeführt werden kann, muss immer zuvor ein Token angefordert werden, dieser Token kann nur einmal verwendet werden.

Path	HTTP-Methode	Beschreibung
/token	GET	Einen neuen Token für einen Upload- oder Downloadaufruf anfordern

Request-Parameter	Art	Typ	Verpflichtend	Beschreibung
-	-	-	-	-

Response-Code	Response-Body	Content-Type	Beschreibung

200	{
	
	"token": <String>,
	
	"validUntil": <String>
	
	}
		application /json	Beinhaltet den Token selbst und das Ablaufdatum.
			
			Das Ablaufdatum *validUntil* wird als Format ISO 8601 (yyyy-MM-dd'T'HH:mm:ss.SSSX) angegeben, Bsp.: 2022-03-22T15:32:00.000 +00:00, in diesem Fall handelt es sich um die UTC-Zeit.
			
			Das Ablaufdatum ermittelt sich wie folgt: Aktuelle Zeit + 5min, das Ergebnis wird bis zur nächsten vollen Minute aufgerundet
			
			Beispiel:
			
			Aktuelle Zeit= 14:32:12
			
			Ablaufdatum = 14:32:12 + 5min = 14:37:12 (Minute aufrunden) =14:38:00

401	siehe oben JSON-Struktur im Fall von Fehlerfällen	application /json	Der Aufrufer hat keine Berechtigung einen Token anzufordern
500	siehe oben JSON-Struktur im Fall von Fehlerfällen	application /json	Technischer Fehler am Server

Beispiel
Request
GET https://<host>:<port>/<service path>/token HTTP/1.1
Response

HTTP/1.1 200
Content-Type: application/json
Content-Length: 103
 	
{
"token" : "2885189e7f3e47558d26765af8809cce",
"validUntil" : "2022-03-22T15:32:00.000+00:00"
}

3.1.2. Upload von Dateien

Über diese Operation ist es möglich Dateien in die ERV-Box hochzuladen. Beim Aufruf handelt es sich um einen HTTP Multipart Request, welches eine JSON-Struktur zur Beschreibung der Dateien und ein oder mehrere Dateien beinhaltet. Damit der Upload erfolgen kann müssen im HTTP-Header folgende Headers gesendet werden:

1. *X-Authorization* der zuvor angeforderte Token Base64 kodiert.
2. *X-Service-Provider* ERV Service Provider Kennung.

Path	HTTP-Methode	Beschreibung
/files	POST	Upload von Metadaten und Dateien

Request-Parameter	Art	Typ	Verpflichtend	Beschreibung
X-Authorization	Header	String	Ja	Beinhaltet den zuvor angeforderten Token: X-Authorization: Bearer <Base64 codierter Token>
X-Service-Provider	Header	String	Ja	Beinhaltet die ERV Service Provider Kennung: X-Service-Provider: CI=<Service Provider Kennung>
metadata	Mime-Part	{ "files": [{ "fileName": <String>, "size": <Number>, "md5": <String>, "crc32": <Number> }] }	Ja	Beinhaltet eine JSON-Struktur, für jede Datei müssen die nachfolgenden Felder mitgegeben werden: **Feld** / **Typ** / **Verpflichtend** / **Anmerkung** fileName / String / Ja / Der Wert muss innerhalb eines Requests eindeutig sein, zusätzlich muss ein Mime-Part existieren dessen Content-Disposition Header das Attribute *filename* mit dem selben Wert besitzt. Folgende Zeichen sind erlaubt: A-Z, a-z, 0-9, - (Bindestrich), _ (Unterstrich), . (Punkt) size / Number / Ja / Größe der Datei in Byte md5 / String / Ja / Hashwert crc32 / Number / Ja / Prüfsumme (Dezimal) Beispiel für die Übertragung im Mime-Part, das Attribut *name* muss den Wert "metadata" besitzen: <Mime-Part boundary> Content-Disposition: form-data; **name="metadata"** Content-Transfer-Encoding: utf-8 Content-Type: application/json; charset=UTF-8 Content-Length: 285 .... <Mime-Part boundary>
file	Mime-Part	Dateinhalt	Ja	Beinhaltet den Dateiinhalt. Im Header Content-Disposition muss mittels *filename* Attribut auf einen Dateinamen in der JSON-Struktur referenziert werden. Beispiel für die Übertragung im Mime-Part, das Attribut *name* muss den Wert "file" besitzen: <Mime-Part boundary> Content-Disposition: form-data; **name="file"**; **filename="Dateinamen.pdf"** Content-Type: application/pdf ..... <Mime-Part boundary>

Response-Code	Response-Body	Content-Type	Beschreibung
200	{ "transferId": <String> }	application/json	Beinhaltet die Transfer-ID, diese kann in einer ERV-Nachricht als Referenz auf die hochgeladenen Dateien verwendet werden.
400	siehe oben JSON-Struktur im Fall von Fehlerfällen	application/json	Request ist fehlerhaft, siehe Feld *errorMessage* für eine genauere Fehlerinformation
401	siehe oben JSON-Struktur im Fall von Fehlerfällen	application/json	Token fehlt oder ist ungültig
415	siehe oben JSON-Struktur im Fall von Fehlerfällen	application/json	Falls kein multipart/form-data als Content-Type gesendet wird
500	siehe oben JSON-Struktur im Fall von Fehlerfällen	application/json	Technischer Fehler am Server

Beispiel mit 2 Dateien

Request

```
POST https://<host>:<port>/<service path>/files HTTP/1.1

X-Authorization: Bearer a2Fqc2Rma2oyMzQyazM0Mmw=
X-Service-Provider: CI=TEST
Content-Type: multipart/form-data; boundary="----=_Part_24_629075685.1648566962748"
MIME-Version: 1.0
Content-Length: 90848347662


------=_Part_24_629075685.1648566962748
Content-Disposition: form-data; name="metadata"
Content-Type: application/json;charset=UTF-8
Content-Transfer-Encoding: binary

{
  "files": [
    {
      "fileName": "Testdatei.pdf",
      "size": 23423423,
      "md5": "2342423423423",
      "crc32": 2342342
    },
    {
      "fileName": "Testvideo.mp4",
      "size": 90824924239,
      "md5": "aekadj2323kaldkksd",
      "crc32": 980982
    }
  ]
}

------=_Part_24_629075685.1648566962748
Content-Disposition: form-data; name="file"; filename="Testdatei.pdf"
Content-Type: application/pdf
Content-Transfer-Encoding: binary

....
------=_Part_24_629075685.1648566962748
Content-Disposition: form-data; name="file"; filename="Testvideo.mp4"
Content-Type: video/mp4
Content-Transfer-Encoding: binary

....
------=_Part_24_629075685.1648566962748--
```

Response

```
HTTP/1.1 200
Content-Type: application/json
Content-Length: 44

{

  "transferId": "kasjdfkj23l4k234"

}
```

3.1.3. Download von Datei-Metadaten

Bevor eine Datei runtergeladen werden kann, muss zuerst über die Metadaten-Abfrage die *fileId* der Datei ermittelt werden. Diese Abfrage liefert alle Datei-Metadaten zu einem Transfer. Im Response ist für jede Datei eine *fileId* enthalten, mit Hilfe dieser ID kann die Datei runtergeladen werden. Für diese Operation ist wie für den Upload oder Download von Dateien ebenfalls ein neuer Token zu verwenden.

Damit der Download von Datei-Metadaten erfolgen kann müssen im HTTP-Header folgende Headers gesendet werden:

1. *X-Authorization* der zuvor angeforderte Token Base64 kodiert.
2. *X-Service-Provider* ERV Service Provider Kennung.

Path	HTTP-Methode	Beschreibung
/files/metadata	GET	Downloaden der Dateimetadaten im JSON-Format

Request-Parameter	Art	Typ	Verpflichtend	Beschreibung
X-Authorization	Header	String	Ja	Beinhaltet den zuvor angeforderten Token: X-Authorization: Bearer <Base64 kodierter Token>
X-Service-Provider	Header	String	Ja	Beinhaltet die ERV Service Provider Kennung: X-Service-Provider: CI=<Service Provider Kennung>
transferId	Query	String	Ja	Kennung des Transfers

Response-Code	Response-Body	Content-Type	Beschreibung
200	{ "files": [{ "fileId": <String>, "fileName": <String>, "size": <Number> }] }	application/json	Beinhaltet eine JSON-Struktur die für jede Datei in dem Transfer die nachfolgenden Felder liefert:
400	siehe oben JSON-Struktur im Fall von Fehlerfällen	application/json	Request ist fehlerhaft, siehe Feld *errorMessage* für eine genauere Fehlerinformation
401	siehe oben JSON-Struktur im Fall von Fehlerfällen	application/json	Token fehlt oder ist ungültig
403	siehe oben JSON-Struktur im Fall von Fehlerfällen	application/json	Aufrufer darf den Transfer nicht runterladen
404	siehe oben JSON-Struktur im Fall von Fehlerfällen	application/json	Unbekannte Transfer-ID
500	siehe oben JSON-Struktur im Fall von Fehlerfällen	application/json	Technischer Fehler am Server

Feld	Typ	Verpflichtend	Anmerkung
fileId	String	Ja	Eindeutige Kennung der Datei, mittels dieser Kennung kann die Datei runtergeladen werden
fileName	String	Ja	Name der Datei
size	Number	Ja	Größe der Datei in Byte

Beispiel

Request

```
GET https://<host>:<port>/<service path>/files/metadata?transferId=<...>  HTTP/1.1

X-Authorization: Bearer a2Fqc2Rma2oyMzQyazM0Mmw=
X-Service-Provider: CI=TEST
Connection: Keep-Alive
```

Response

```
HTTP/1.1 200
Content-Type: application/json
Content-Length: 237

{
  "files": [
    {
      "fileId": "kwrk2342",
      "fileName": "Testvideo.mp4",
      "size": 234234243
    },
    {
      "fileId": "mji23kaa",
      "fileName": "Testvideo_2.mp4",
      "size": 93453245
    }
  ]
}
```

3.1.4. Download einer Datei

Nach dem mittels der Metadaten-Abfrage (siehe oben) die File-ID (*fileId*) ermittelt wurde, kann die entsprechende Datei mit dieser Operation runtergeladen werden. Die Datei wird samt einer JSON-Datei welche Metadaten zu der Datei beinhaltet, in einer ZIP-Datei ausgeliefert, somit befinden sich immer 2 Dateien in der ZIP-Datei.

Damit der Download einer Datei erfolgen kann müssen im HTTP-Header folgende Headers gesendet werden:

1. *X-Authorization* der zuvor angeforderte Token Base64 kodiert.
2. *X-Service-Provider* ERV Service Provider Kennung.

Path	HTTP-Methode	Beschreibung
/files	GET	Download einer Datei samt Metadaten in einer ZIP-Datei

Request-Parameter	Art	Typ	Verpflichtend	Beschreibung
X-Authorization	Header	String	Ja	Beinhaltet den zuvor angeforderten Token: X-Authorization: Bearer <Base64 kodierter Token>
X-Service-Provider	Header	String	Ja	Beinhaltet die ERV Service Provider Kennung: X-Service-Provider: CI=<Service Provider Kennung>
transferId	Query	String	Ja	Kennung des Transfers
fileId	Query	String	Ja	Kennung der Datei die runtergeladen werden soll

Response-Code	Response-Body	Content-Type	Beschreibung
200	ZIP-Dateiinhalt	application/zip	Beinhaltet eine ZIP-Datei mit folgendem Inhalt: ▪ Metadaten-Datei im JSON-Format (metadata.json) ▪ Angeforderte Datei
400	siehe oben JSON-Struktur im Fall von Fehlerfällen	application/json	Request ist fehlerhaft, siehe Feld *errorMessage* für eine genauere Fehlerinformation
401	siehe oben JSON-Struktur im Fall von Fehlerfällen	application/json	Token fehlt oder ist ungültig
404	siehe oben JSON-Struktur im Fall von Fehlerfällen	application/json	Unbekannte Transfer- oder File-ID, nähere Informationen siehe Fehlertext
500	siehe oben JSON-Struktur im Fall von Fehlerfällen	application/json	Technischer Fehler am Server

Beispiel

Metadaten-Datei metadata.json

```
{
"fileId": "kwrk2342",
"fileName": "Testvideo.mp4",
"size": 23423424,
"crc32": 3632233996,
"md5": "098f6bcd4621d373cade4e832627b4f6",
"contentType": "video/mp4"
}
```

Beispiel

Request

```
GET https://<host>:<port>/<service path>/files?transferId=<...>&fileId=<...>  HTTP/1.1

X-Authorization: Bearer a2Fqc2Rma2oyMzQyazM0Mmw=
X-Service-Provider: CI=TEST
Connection: Keep-Alive
```

Response

```
HTTP/1.1 200

Content-Type: application/zip



<Inhalt der ZIP-Datei>
```