Anhang 3 der Verordnung vom 8. Dezember 2017 des Eidgenössischen Justiz- und Polizeidepartements (EJPD) über die Erstellung elektronischer öffentlicher Urkunden und elektronischer Beglaubigungen (EÖBV-EJPD; SR 211.435.11)

Technische Anforderungen zur Vermittlung des Zugriffs auf das UPReg

Version: 2

Inkrafttreten: 15.03.2022

Inhaltsverzeichnis

1 Allgemeines

1.1 Grundlage

Das vorliegende Dokument bildet Anhang 3 der Verordnung des EJPD über die Erstellung elektronischer öffentlicher Urkunden und elektronischer Beglaubigungen (EÖBV-EJPD; SR 211.435.11). Es stützt sich auf Artikel 10 Absatz 4 und Artikel 20 der Verordnung über die Erstellung elektronischer öffentlicher Urkunden und elektronischer Beglaubigungen (EÖBV; SR 211.435.1) sowie auf Artikel 9 Absatz 1 und Artikel 10 EÖBV-EJPD.

1.2 Gegenstand

Der Kontakt zwischen dem von der Urkundsperson verwendeten Informatiksystem und dem Schweizerischen Register der Urkundspersonen (UPReg) kann durch Dritte vermittelt werden (Art. 10 Abs. 4 EÖBV). UPReg bietet Dritten die technische Möglichkeit, Zulassungsbestätigungen (Art. 2 Bst. a EÖBV) abzurufen (Art. 9 Abs. 1 EÖBV-EJPD). Dritte bedürfen dabei einer Bewilligung des Bundesamtes für Justiz (BJ).

Das vorliegende Dokument legt die technischen Anforderungen fest, die zur Erteilung der Bewilligung nach Artikel 20 Absatz 1 Buchstabe b EÖBV-EJPD erfüllt sein müssen. Darüber hinaus enthält das vorliegende Dokument auch Anforderungen an die UPReg-Schnittstelle zur Ausgabe von Zulassungsbestätigungen an Urkundspersonen.

1.3 Geltungsbereich

Die Anforderungen richten sich an Dritte, die den Zugriff auf das UPReg für den Abruf von Zulassungsbestätigungen zu vermitteln beabsichtigen.

1.4 Gegenstand der Prüfung im Rahmen des Bewilligungsverfahrens

Das Gesuch um Bewilligung, den Zugriff auf das UPReg zu vermitteln, muss Angaben enthalten, wie die folgenden Vorgaben erfüllt sind (Art. 10 Abs. 2 EÖBV-EJPD):

1. den Nachweis der Erfüllung der Anforderungen an die Informationssicherheit nach Kapitel 2;
2. den Nachweis der Erfüllung der Anforderungen an den Dienst und die Dritte nach Kapitel 3;
3. den Nachweis der Erfüllung der Anforderungen an die technische Implementierung nach den Kapiteln 5 – 11.

Das BJ prüft die Erfüllung der organisatorischen, betrieblichen und technischen Anforderungen. Vor der produktiven Inbetriebnahme ist die Funktionsfähigkeit in der Praxis dem BJ anhand von Beispielen von öffentlichen elektronischen Urkunden in einer Testumgebung nachzuweisen. Das BJ stellt dazu einen Zugang auf ein Testsystem zur Verfügung.

2 Anforderungen an die Informationssicherheit

2.1 Anforderungen an Behörden

[1] Die Anforderungen an den Betrieb durch eine Behörde bzw. deren Einhaltung bilden nicht Gegenstand des Bewilligungsverfahrens nach Artikel 20 Absatz 1 Buchstabe b EÖBV und sind folglich auch nicht Gegenstand des vorliegenden Anhangs.

[2] Die Anforderungen an den Betrieb durch eine Behörde richten sich nach den jeweils für die gesuchstellende Behörde anwendbaren Bestimmungen. Dementsprechend liegt die Verantwortung für den Betrieb durch eine Behörde bei der betreffenden Organisation.

[3] Zieht die gesuchstellende Behörde für den technischen Betrieb private Leitungserbringer bei, müssen die beigezogenen privaten Leitungserbringer die Anforderungen nach Abschnitt 2.3 erfüllen. Die gesuchstellende Behörde trifft im Rahmen der Instruktion des Bewilligungsverfahrens eine Mitwirkungspflicht und sie stellt sicher, dass auch die beigezogenen privaten Leitungserbringer an der Instruktion im selben Umfang mitwirken.

2.2 Anforderungen an Privatunternehmen

[1] Ist die Dritte ein Privatunternehmen, ist die Informationssicherheit durch Einrichtung, Umsetzung, Betrieb, Überwachung, Überprüfung, Aufrechterhaltung und Verbesserung eines Informationssicherheitsmanagementsystems (ISMS) nach ISO/IEC 27001:2013, Information technology – Security techniques – Information security management systems – Requirements[1] sicherzustellen.

[2] Die Wirksamkeit und Angemessenheit des ISMS ist durch eine Zertifizierung nach ISO/IEC 27001:2013 nachzuweisen. Das dazugehörige Zertifikat ist durch eine von der Schweizerischen Akkreditierungsstelle (SAS) akkreditierte Zertifizierungsstelle auszustellen. Alle Elemente des Dienstangebots müssen im Anwendungsbereich des zertifizierten ISMS liegen.

[3] Wird eine neue Ausgabe der Norm ISO/IEC 27001:2013 publiziert, muss spätestens nach Ablauf der durch die SAS festgelegten Übergangsfrist eine gültige Zertifizierung des ISMS nach dieser neuen Ausgabe nachgewiesen werden. Alle Elemente des Dienstangebots der Zustellplattform müssen weiterhin im Anwendungsbereich des zertifizierten ISMS liegen.

2.3 Ausgliederung des technischen Betriebs an private Leistungserbringer

[1] Die Dritten kann den technischen Betrieb der Plattform ganz oder teilweise auch an private Leistungserbringer übertragen. Sie behält dabei die technische, administrative, rechtliche und Führungsverantwortung.

[2] Der private Leistungserbringer erfüllt die folgenden Anforderungen:

- die Informationssicherheit des technischen Betriebs der Plattform ist durch ein Informationssicherheitsmanagementsystem (ISMS) gemäss ISO/IEC 27001:2013 sichergestellt;
- der Anwendungsbereich dieses ISMS umfasst: alle Organisationseinheiten, in denen der technische Betrieb ganz oder teilweise abgewickelt wird; alle Personen (Beschäftigte, Auftragnehmer und Beschäftigte von Lieferanten), die für den Betrieb zuständig oder verantwortlich sind; alle technischen Einrichtungen und Geräte, die für den Betrieb erforderlich sind; alle Räumlichkeiten, die für den technischen Betrieb erforderlich sind;
- der technische Betrieb erfolgt auf Schweizer Territorium;
- der private Leistungserbringer untersteht ausschliesslich Schweizer Rechtsprechung;
- das ISMS ist durch eine Zerfifizierungsstelle zertifiziert, die durch die Schweizerische Akkreditierungsstelle (SAS) anerkannt ist.

[3] Alternativ kann der technische Betrieb nach einer Bestätigung der Datenschutzkonformität durch den Eidgenössischen Datenschutz- und Öffentlichkeitsbeauftragten (EDÖB), auch ganz oder teilweise an einen privaten Leistungserbringer ausgegliedert werden, der auf einem Territorium operiert und einer Rechtsprechung untersteht, die durch den EDÖB als aequivalent beurteilt wird. Die komplette für den

1 Die aufgeführte Norm kann bezogen werden bei der Schweizerischen Normen-Vereinigung (SNV), Sulzeralle 70, 8404 Winterthur, www.snv.ch.

Betrieb erforderliche Infrastruktur muss durch diesen Betreiber auf diesem Territorium betrieben werden. Die Prüfung der Datenschutzkonformität durch den EDÖB wird vom BJ im Rahmen des Bewilligungsverfahrens nach Artikel 20 Absatz 1 Buchstabe b EÖBV veranlasst.

[4] Die Gesamtverantwortung für alle Tätigkeiten, die für den Betrieb erforderlich sind oder mit diesem in Zusammenhang stehen, wird weiterhin durch die Dritte sichergestellt.

3 Anforderungen an das IT Service Management

3.1 Grundanforderungen

[1] Für den zuverlässigen Betrieb der Plattform muss nachgewiesen werden, dass Betriebsprozesse dokumentiert, eingeführt, betrieben, überwacht, in geplanten Abständen überprüft, aufrechterhalten und verbessert werden.

[2] Die Betriebsprozesse haben sich an der Norm ISO/IEC 20000-1:2018 Information technology — Service management — Part 1: Service management system requirements oder einer vergleichbaren Norm zu orientieren. Eine entsprechende Zertifizierung ist erwünscht, aber nicht erforderlich.

[3] Zusätzlich muss ein professioneller Service Desk eingeführt, betrieben, in geplanten Abständen überprüft, aufrechterhalten und verbessert werden.

3.2 Verfügbarkeit

[1] Der Dienst muss an allen Tagen rund um die Uhr zur Verfügung stehen. Allfällige Servicefenster sind zwischen 00:15 Uhr und 07:00 Uhr nach Schweizer Zeit oder an Wochenenden zu planen. Sie sind auf der Webseite des Dritten mindestens 72 Stunden im Voraus zu veröffentlichen.

[2] Die Verfügbarkeit des Dienstes wird protokolliert und das Protokoll veröffentlicht.

3.3 Informationen für Benutzerinnen und Benutzer

Die Dritte muss auf ihrer Webseite in einer für den technischen Laien verständlichen Form die folgenden wesentlichen Merkmale veröffentlichen:

a. die Architektur;

b. die Verfahren der Zugriffssteuerung;

c. die verwendeten kryptografischen Verfahren und Systeme.

4. Glossar

Begriff / Abkürzung	Bedeutung
Authenticator	Ein FIDO-Gerät. Ein kryptografisches Gerät (Security-Token) mit einer Schnittstelle zum Computer (USB A, USB C oder NFC) und einer berührungssensitiven Kontaktfläche für die Auslösung einer Schlüsselübermittlung im Authentisierungsprozess.
Client	Eine Anwendung für das Anbringen einer Zulassungsbestätigung auf ein qualifiziert signiertes Dokument.
Cygillum	Cygillum ist ein als eine auf dem Desktop installierte Webanwendung konzipierter Client. Die Anwendung verfügt über ein in einem Browser angezeigtes Front-End und ein lokales Back-End.
FIDO	Standard der FIDO («Fast IDentity Online») -Allianz und des W3C, der eine starke Authentifizierungslösung mittels Public-Key-Verfahren im Web realisiert.
Happy Path	Bezeichnung des einfachsten Szenarios, in welcher ein Algorithmus funktioniert, ohne Ausnahmen oder Fehlerzustände.

REST	Representational State Transfer ist ein Paradigma für die Softwarearchitektur von Webservices.
UUID	Universally Unique Identifier. Eine 128-Bit lange Zahl.
WebAuthn	WebAuthn ist ein vom World Wide Web Consortium (W3C) veröffentlichter Standard für eine direkte Authentifikation mittels Public-Key-Verfahren im Webbrowser.

5. Grundsätzliches zum Authentisierungsverfahren

Als Authentisierungsverfahren wird WebAuthn mit FIDO-Geräten eingesetzt. Es werden diejenigen Versionen unterstützt, die von den jeweils aktuellen Browsern unterstützt werden. Für die Ausstellung einer Zulassungsbestätigung muss der entsprechende Client (Cygillum oder ein System eines Dritten) die Urkundspersonen entweder mittels WebAuthn UPReg authentisieren oder einen von der Urkundsperson signierten Antrag (eine signierte XML-Datei) vorlegen.

6. Grundsätzliches zum REST-Webservice

6.1. Protokolle

Die Schnittstelle ist als REST Webservice definiert. Für das Messaging wird HTTP 1.1 und als Datenaustauschformat JSON verwendet. Die Kommunikation zwischen Client und Server des Webservice läuft ausschliesslich über HTTPS.

6.2. Serverzertifikat

Zur Absicherung des HTTPS-Protokolls gegen Man-in-the-Middle-Angriffe mit gefälschten, jedoch von einer anerkannten Zertifizierungsstelle (certificate authority) signierten Zertifikaten, ist es in der Verantwortung des Clients zu überprüfen, ob der Verbindungsaufbau zum «richtigen» Server erfolgt.
Dabei muss der Client prüfen, ob der Aussteller des Server-Zertifikates mit einer im DNS CAA Record für die Domain upreg.ch aufgeführten Zertifizierungsstelle übereinstimmt und ob der DN des Server-Zertifikates `www.upreg.ch` ist.

6.3. Allgemeine HTTP Status-Codes

Der Web-Service liefert die folgenden Standard HTTP Status Codes zurück:

Status Code	Bedeutung
200	Die Operation ist vom Server verarbeitet worden.
400	Bad request (Syntaxfehler). Dies tritt z.B. bei einem nicht eingegebenen Parameter auf oder wenn der Parameter fehlerhaft ist (Format, Typ).
403	Forbidden (Zugang verboten). Es darf nicht auf die Ressource zugegriffen werden.
404	Not found (Ressource existiert nicht). Z.B. wenn die URL falsch ist.
405	Method not allowed. Wird zurückgeliefert, wenn für eine der Operationen die falsche HTTP-Methode verwendet wird.
408	Request Timeout. Die Session existiert nicht oder nicht mehr.
409	Conflict. Mehrere Personen mit demselben Zertifikat in einem Register.
415	Unsupported Media Type. Wenn im Header Content-Type nicht application/json angegeben ist.
429	Too Many Requests. Zu viele offene Transaktionen für diese IP-Adresse.
500	Internal Server Error. Wird zurückgesendet, wenn serverseitig ein Fehler aufgetreten ist.

7. Ausstellung der Zulassungsbestätigung

7.1. Einleitung

Die Ausstellung der Zulassungsbestätigung für ein Dokument besteht aus der Vogelperspektive betrachtet aus den folgenden Schritten, die weiter unten näher erläutert werden:

- Initiieren des Vorgangs (startTransactions).
- Prüfung der Berechtigung (claim oder authenticate).
- Erzeugung der bildlichen Repräsentation der Zulassungsbestätigung (RT1-Generate).
- Signieren der Zulassungsbestätigung (RT2-Sign).

7.2. startTransactions

Ein Client initiiert den Vorgang zur Ausstellung einer Zulassungsbestätigung für ein oder mehrere konkrete Dokumente durch Aufruf der Operation startTransaction. UPReg erzeugt als Folge die gewünschte Anzahl Transaktionen, welche aus je zwei zufälligen Tokens bestehen. Das eine Token (Auth-Token) dient der Authentisierung für den nachfolgenden Schritt der Überprüfung der Berechtigung. Das andere Token (ZB-Token) dient der Ausstellung der Zulassungsbestätigung in den beiden Schritten RT1-Generate und RT2-Sign. Ein Token kann nur einmal verwendet werden. Beide Tokens haben eine begrenzte Gültigkeit. Als Antwort erhält der Client für jedes Dokument ein Token-Paar.

7.3. Prüfung der Berechtigung

UPReg stellt Zulassungsbestätigungen nur für registrierte Urkundspersonen aus. Die Prüfung der Berechtigung kann von der Client-Software auf zwei unterschiedliche Arten durchgeführt werden:

- Mit Übermittlung eines von der Urkundsperson digital signierten Antrags: claim
- Mit Authentisierung der Urkundsperson bei UPReg mit dem WebAuthn-Protokoll: authenticate

7.3.1. Prüfung der Berechtigung mit digitaler Signatur

Der Client erstellt ein XML-Dokument, das die folgenden Informationen enthält:

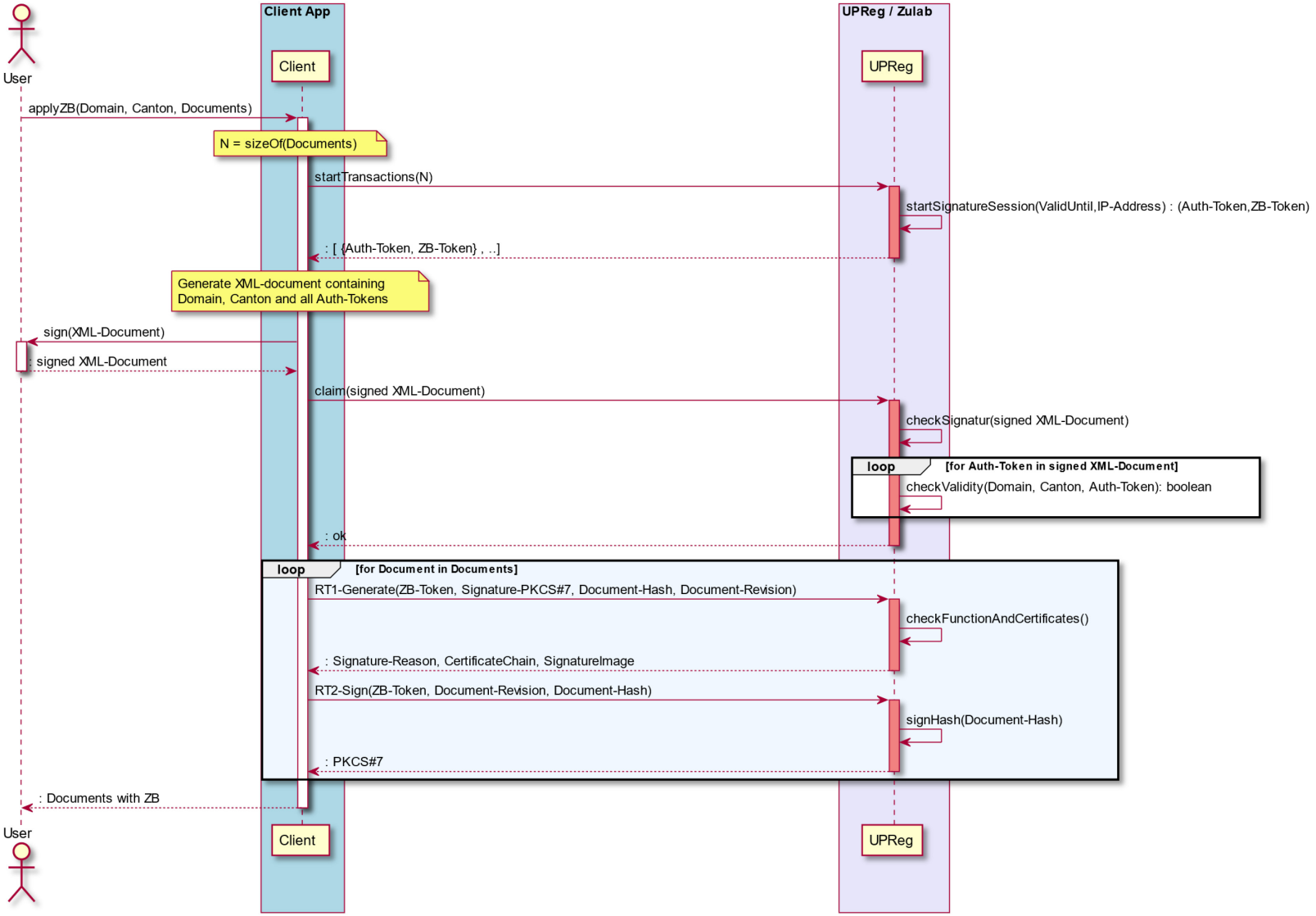
- Den Kanton und die Domäne, für welche die Zulassungsbestätigung angefordert wird;
- Für jedes Dokument das von startTransactions zurückgegebene Auth-Token.

Das XML-Dokument lässt der Client von der Urkundsperson mit einem qualifizierten Zertifikat digital signieren. Anschliessend ruft der Client die claim Operation von UPReg auf und übermittelt das signierte XML-Dokument.

UPReg prüft, ob das für die Signatur verwendete Zertifikat hinterlegt ist und ob die Auth-Tokens zu einer noch aktiven Session gehören. Falls beide Prüfungen erfolgreich waren, wird die zugehörige Urkundsperson mit der Session assoziiert und die zugehörige ZB-Tokens können in den nachfolgenden Schritten verwendet werden.

Das Verfahren erlaubt, Zulassungsbestätigungen für mehrere Dokumente der gleichen Urkundsperson in einem Stapelverfahren abzurufen.

Im UML Sequenzdiagramm in Abbildung 1 wird der Happy Path der Ausstellung einer Zulassungsbestätigung mit digitaler Signatur beschrieben. Wie der Client die Signatur der Urkundsperson auf dem XML-Dokument einholt, ist für den Ablauf der Ausstellung der Zulassungsbestätigung nicht von Interesse.


User
Client App
Client
UPReg / Zulab
UPReg
applyZB(Domain, Canton, Documents)
N = sizeOf(Documents)
startTransactions(N)
startSignatureSession(ValidUntil,IP-Address) : (Auth-Token,ZB-Token)
: [{Auth-Token, ZB-Token} , ..]
Generate XML-document containing
Domain, Canton and all Auth-Tokens
sign(XML-Document)
: signed XML-Document
claim(signed XML-Document)
checkSignatur(signed XML-Document)
loop
[for Auth-Token in signed XML-Document]
checkValidity(Domain, Canton, Auth-Token): boolean
: ok
loop
[for Document in Documents]
RT1-Generate(ZB-Token, Signature-PKCS#7, Document-Hash, Document-Revision)
checkFunctionAndCertificates()
: Signature-Reason, CertificateChain, SignatureImage
RT2-Sign(ZB-Token, Document-Revision, Document-Hash)
signHash(Document-Hash)
: PKCS#7
: Documents with ZB
User
Client
UPReg

7.3.2. Prüfung der Berechtigung mit WebAuthn-Protokoll

Für die Authentisierung der Urkundsperson bei UPReg wird das WebAuthn Protokoll unter Verwendung des FIDO Authenticators der Urkundsperson verwendet. Die Interaktion muss folglich zwingend im Webbrowser erfolgen. Der Client verwendet den Webbrowser der Urkundsperson, um die authenticate-Operation von UPReg aufrufen zu lassen.

Beim Aufruf der Authentisierung müssen die von der startTransactions-Operationen zurückgegebenen Authentisierungs-Tokens (Auth-Token) übermittelt werden. Das aufrufende System kann optional eine Provider-ID, eine eigene Session-ID und einen HTTP-Port mitgeben, die für den Rücksprung von UPReg auf das aufrufende System verwendet werden sollen.

Die Urkundsperson wird zur Authentisierung mit dem Webbrowser auf UPReg weitergeleitet und authentisiert sich dort mit Benutzernamen, Passwort und dem FIDO Authenticator. UPReg prüft zuerst Benutzernamen, Passwort und FIDO Authenticator. Anschliessend wird anhand der Auth-Tokens geprüft, ob die Session noch aktiv ist. Waren alle Prüfungen erfolgreich, wird die authentisierte Urkundsperson mit der Session assoziiert und die zugehörigen ZB-Tokens können in den nachfolgenden Schritten verwendet werden.

Nach dem Authentisierungsvorgang wird die Urkundsperson an die durch die Provider-ID bestimmte Rücksprungsadresse des aufrufenden Systems zurückgeleitet. Dabei wird das Resultat des Authentisierungsvorgangs übermittelt.

Das Authentisierungsverfahren erlaubt, Zulassungsbestätigungen für mehrere Dokumente der gleichen Urkundsperson in einem Stapelverfahren abzurufen.

Im UML Sequenzdiagramm in Abbildung 2 wird der Happy Path der Ausstellung einer Zulassungsbestätigung mit Authentisierung mittels WebAuthn beschrieben. Der Ablauf wird aus der Perspektive der Anwendung Cygillum beschrieben, die vollständig lokal auf dem Arbeitsplatz einer Urkundsperson ausgeführt wird. Die Anwendung verfügt über ein lokal installiertes Back-End (eine lokale Web-Anwendung) und einen Front-End im Browser. Ein Client eines Dritten wird unter Umständen das Back-End auf einem Server, also nicht lokal auf dem Arbeitsplatz der Urkundsperson betreiben.

Im nachfolgenden Sequenzdiagramm sind die Aktivierungsbalken (Focus of Control), die Webservice-Aufrufe betreffen, in Weiss gehalten. Rote Aktivierungsbalken betreffen Interaktionen der mit WebAuthn ausgeführten Authentisierung im Webbrowser.

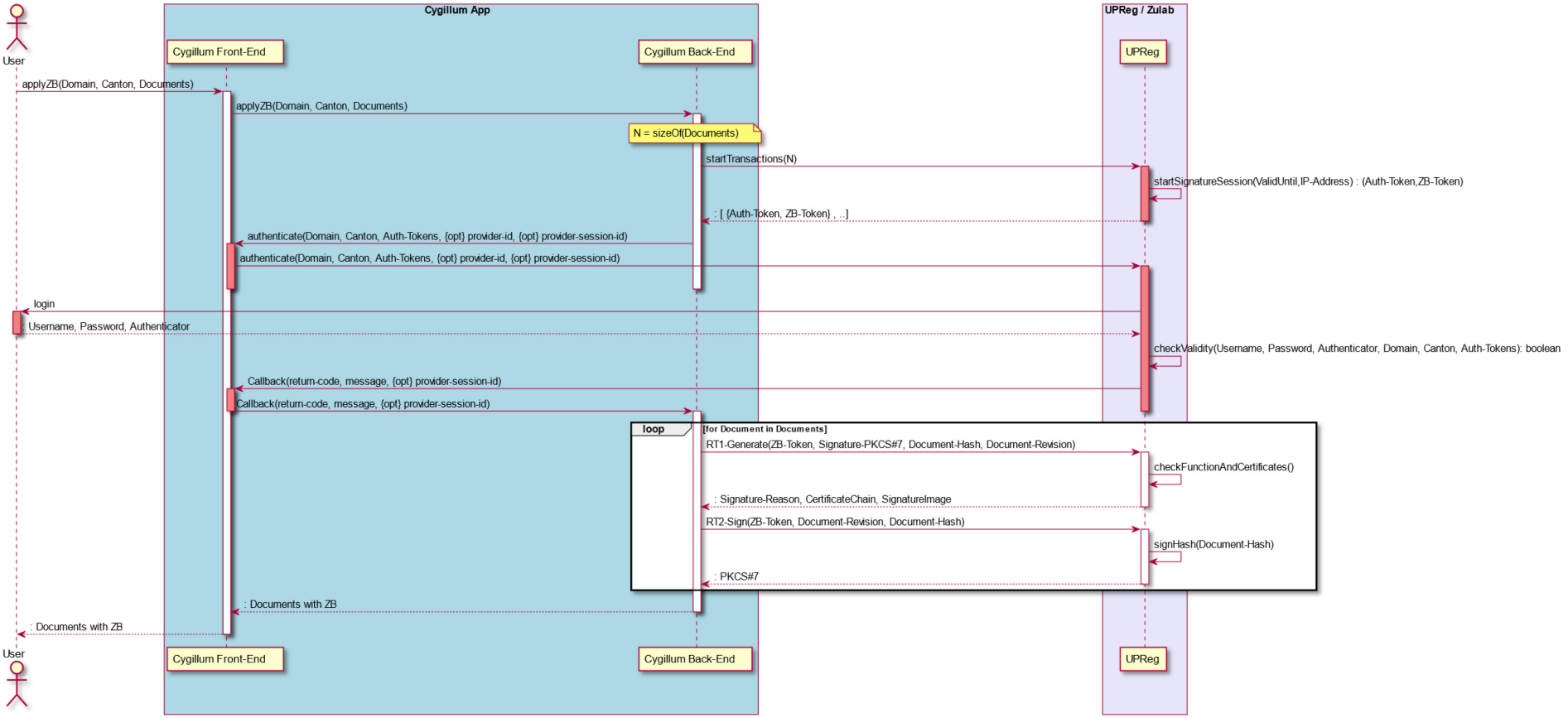


Abbildung 2: Zulassungsbestätigung mit WebAuthn

7.4. RT1-Generate

Nach erfolgreicher Prüfung der Berechtigung ruft der Client den WebService RT1-Generate auf. Diesem wird das ZB-Token, die Signatur in Form eines PKCS#7, der mit der Signatur signierte Hash des Dokuments sowie dessen Document-Revision übergeben.

UPReg prüft die Session anhand des ZB-Tokens und die der Urkundsperson zugewiesene Funktion anhand des Zertifikats in der Signatur, dem Kanton und der Domäne. Als Antwort dieser Operation wird die Signature-Reason, die Zertifikatskette und das einzubettende Signaturbild zurückgeliefert.

7.5. RT2-Sign

Hat der Client die von RT1-Generate gelieferten Daten verarbeitet, wird die Operation RT2-Sign aufgerufen. Dieser wird das ZB-Token, die Document-Revision des signierten Dokuments (Document-Revision aus RT1-Generate plus eins) und der zu signierende Dokument-Hash mitgegeben. Zwischen der Signatur der Urkundsperson und der Signatur der Zulassungsbestätigungen dürfen keine Revisionen (z.B. LTV-Informationen o.ä.) in das PDF-Dokument eingebettet werden.

UPReg erzeugt die Signatur für die Zulassungsbestätigung und schickt als Resultat der Operation die Signatur in Form einer PKCS#7 Struktur an den Client zurück. Der Client bettet die Signatur in das Dokument ein.

8. Authentisierung mit WebAuthn

Die Authentisierung der Urkundspersonen mit WebAuthn zwischen den beiden Webservice-Aufrufen *startTransactions* und *RT1-Generate* erfolgt im Webbrowser. Dazu werden von UPReg der Username, das Passwort und der von der Urkundsperson im Registrierungsprozess auf UPReg hinterlegte öffentliche Schlüssel des FIDO Authenticators geprüft.
Das aufrufende System muss für diesen Authentisierungsvorgang die Kontrolle an den Webbrowser übergeben und wird nach erfolgter Authentisierung die Kontrolle für die weiteren Schritte wiedererlangen.

8.1. Request für die Authentisierung

Das Back-End des aufrufenden Systems muss den Webbrowser der Urkundsperson mittels HTTP POST Request auf die folgenden URL leiten.

```
https://[host]/zulab/authenticate
```

`[host]` bezeichnet dabei die UPReg-Instanz, die angesprochen werden soll. Test, Vorproduktion, Produktion haben unterschiedliche Host-Namen.

Folgender Request-Header muss gesetzt werden:

```
Content-Type: text/plain
```

Das zu übermittelnde JSON-Objekt ist in ein Formularfeld namens `data` einzubetten.

Beispiel:

```
<form action="https://www.upreg.ch/zulab/authenticate"
      method="POST"
      enctype="text/plain">

    <input type="textarea" id="data" name="data" value="">
    <button type="submit" id="submit">Send</button>
  </form>
```

Die Attribute des zu übermittelnden JSON-Objektes sind wie folgt:

JSON Attribut	Format	Beschreibung
auth-tokens	Array von String	Erforderlich. Die als Resultat von *startTransactions* erhaltenen Authentisierungs-Tokens. Der Array darf maximal 100 Tokens enthalten.
domain	String	Kürzel der Domäne, in welcher die Urkundsperson registriert ist. Zulässige Werte sind der Kantons- und Domänenliste zu entnehmen.
canton	String	Kürzel des Kantons, in welchem die Urkundsperson registriert ist. Zulässige Wert sind der Kantons- und Domänenliste zu entnehmen.
provider-id	String	Optional. Kennung des aufrufenden Systems, für die in UPReg eine entsprechende Rücksprungsadresse (ein URL) hinterlegt ist. Ist das Attribut nicht vorhanden, wird Cygillum als aufrufendes System angenommen
port	Number	Optional. Zulässige Werte: 0 bis 65 535 (= 2^{16} - 1). http-Port der Rücksprungadresse. Die Angabe des http-Ports ist erforderlich, falls das aufrufende System dynamische http-Ports verwendet. Ist das Attribut nicht vorhanden, wird der Port 443 (SSL) angenommen.
provider-session-id	String	Optional. Eine frei wählbare, ggf. URL-codierte Zeichenfolge, mit der das aufrufende System beim Rücksprung eine Session des Benutzers wieder zuordnen kann. Ist das Attribut nicht vorhanden, wird in der Antwort das gleichnamige Attribut weggelassen.

Beispiel:

```
{
    "auth-tokens": ["942258e6-b331-49d7-9232-d2a838e5d656"],
    "domain": "upreg",
    "canton": "SO",
    "provider-id": "terravis",
    "port": 443,
    "provider-session-id": "eb052661-20bc-45c3-9479-4b6420862413"
}
```

8.2. Webseite für die Authentisierung

Durch den Request für die Authentisierung gelangt man auf die Webseite für die Authentisierung. Darauf wird der Kanton, die Domäne und die Anzahl freizugebender Zulassungsbestätigungen angezeigt. Der Benutzer gibt darin seinen Benutzernamen und das Passwort ein. Daraufhin wird sein zweiter Faktor (WebAuthn-Token) geprüft. Falls der Benutzer authentisiert werden kann, kann er zurück zum Zulassungsbestätigungs-Provider (z.B. Cygillum) navigieren. Ein Abbruch des Vorgangs, inklusive Rücksprung auf den Zulassungsbestätigungs-Provider ist jederzeit möglich.

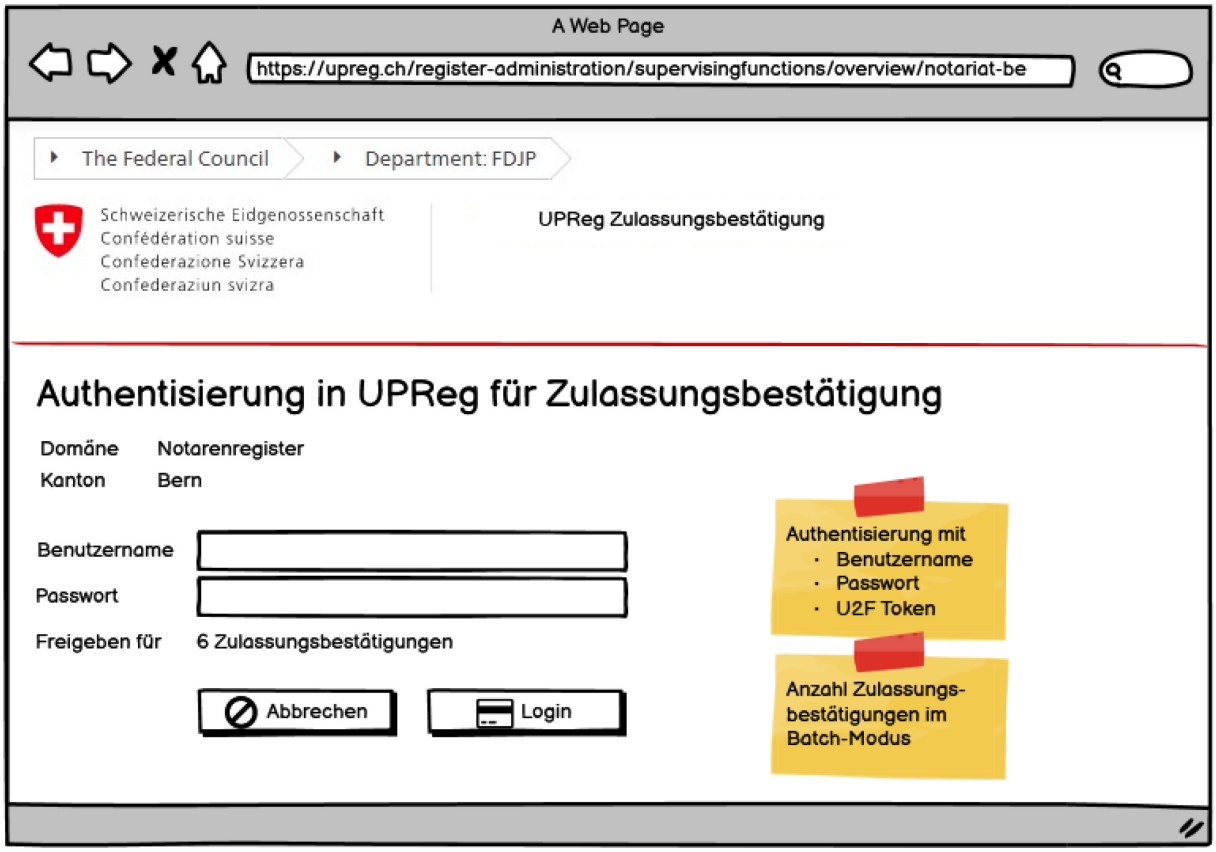


Abbildung 3: Mockup der Webseite für die Authentisierung

8.3. Rücksprung

Der Ausgang der Authentisierung wird dem aufrufenden System mit einem http POST Request auf die unter der Provider ID hinterlegte URL übermittelt.

Folgender Request-Header muss gesetzt werden:

```
Content-Type: text/plain
```

Das zu übermittelnde JSON-Objekt ist in ein Formularfeld namens `data` einzubetten.

Beispiel:

```
<form action="https://localhost:1234/cygillum/landing"
      method="POST"
      enctype="text/plain">

    <input type="textarea" id="data" name="data" value="">
    <button type="submit" id="submit">Send</button>
  </form>
```

Die Attribute des JSON-Objektes sind wie folgt:

JSON- Attribut	Format	Beschreibung
provider-session-id	String	Optional. Die vom aufrufenden System im Authentisierungs-Aufruf übermittelte eigene Session-ID. Das Attribut ist vorhanden, wenn beim Aufruf das gleichnamige Attribut mitgegeben wurde.

<table>
<tr><th>JSON- Attribut</th><th>Format</th><th>Beschreibung</th></tr>
<tr><td>return-code</td><td>String</td><td>Resultatcode der Authentisierung mit folgender Bedeutung:
<table>
<tr><th>return-code</th><th>Beschreibung</th></tr>
<tr><td>1</td><td>Die Urkundsperson konnte erfolgreich authentisiert werden.</td></tr>
<tr><td>2</td><td>Die Urkundsperson hat den Vorgang abgebrochen.</td></tr>
<tr><td>3</td><td>Während der Authentisierung ist ein Fehler aufgetreten.
Eine Problembeschreibung wird auf der Authentisierungs-Webseite angezeigt, jedoch nicht an den Client eskaliert.</td></tr>
</table></td></tr>
<tr><td>message</td><td>String</td><td>Menschenlesbare Meldungen mit allfällig zusätzlichen Informationen zum return-code.</td></tr>
</table>

Beispiel:

```
{
    "provider-session-id": "eb052661-20bc-45c3-9479-4b6420862413",
    "return-code": "1",
    "message": "You've made my day"
}
```

9. REST Interface für Zulassungsbestätigung

9.1. Methode *startTransactions*

Um eine Transaktion für das Anbringen einer Zulassungsbestätigung zu starten, ruft der Client die Operation *startTransactions* auf. Der Client gibt dabei an, für wie viele Dokumente er eine Zulassungsbestätigung möchte.
Auf UPReg wird mit diesem Aufruf eine Session gestartet, welche maximal zehn Minuten lang gültig ist. Das Resultat der Operation ist ein Array von Token-Paaren, die für die nachfolgende Prüfung der Berechtigung (Auth-Token) und die Aufrufe für die Ausstellungen der Zulassungsbestätigung selbst benötigt werden (ZB-Token).

9.1.1. Aufruf

```
POST /CONTEXT-ROOT/zulab/startTransactions
```

Die Attribute des JSON-Objektes im Request-Body sind wie folgt:

JSON Attribut	Format	Beschreibung
count	Number	Anzahl der Dokumente, für welche eine Zulassungsbestätigung erstellt werden soll. 1 ≤ count ≤ 100

Beispiel der JSON Struktur:

```
{
   "count": "2"
}
```

Erforderliche Request Header:

```
Content-Type: application/json
Accept: application/json
```

9.1.2. Antwort

Im Erfolgsfall wird der HTTP Status-Code 200 zurückgeliefert und im Response-Body steht als Antwort ein Array von *count* JSON-Objekten. Im Request-Header ist der MIME-Type JSON angegeben.

```
Content-Type: application/json
```

Die Attribute eines einzelnen JSON-Objektes im Array sind wie folgt:

JSON Attribut	Format	Beschreibung
auth-token	String	Authentisierungs-Token. Eine UUID Version 4 (zufällig generierte UUID). Wird bei der Prüfung der Berechtigung benötigt.
zb-token	String	Token für die Ausstellung der Zulassungsbestätigung. Eine UUID Version 4 (zufällig generierte UUID). Wird für die Webservice-Aufrufe rt1-generate und rt2-sign benötigt.

Beispiel der zurückgelieferten JSON-Struktur:

```
[
    {
        "auth-token": "ac13184a-2bbf-4a55-b9b3-2374b48a3655",
        "zb-token": "3d588980-bc3b-4e6d-b305-212eb2f844d9"
    },
    {
        "auth-token": "4b7d3c60-bece-4f90-9a71-aa8c18783de1",
        "zb-token": "78138f1f-de30-4186-830d-17e06385ca51"
    }
]
```

9.2. Methode *claim*

Dient der Prüfung der Berechtigung mit einem von der Urkundsperson digital signiertem Antrag. UPReg prüft, ob das für die Signatur verwendete Zertifikat hinterlegt ist und ob jeder übermittelte Auth-Token zu einer noch aktiven Transaktion gehören. Falls beide Prüfungen erfolgreich waren, wird die zugehörige Urkundsperson mit den Transaktionen assoziiert und der zugehörige ZB-Token kann für den Aufruf von rt1-generate und rt2-sign verwendet werden.

Das Verfahren erlaubt, Zulassungsbestätigungen für mehrere Dokumente der gleichen Urkundsperson in einem Stapelverfahren abzurufen.

9.2.1. Aufruf

```
POST /CONTEXT-ROOT/zulab/claim
```

Im POST Request wird ein digital signiertes XML-Dokument übermittelt, dass dem XML-Schema http://www.upreg.ch/claim/1 (siehe Kapitel 11.1) entspricht. Das XML-Dokument muss mit einem im UPReg registrierten qualifizierten Zertifikat einer Urkundsperson signiert sein. Ein Zeitstempel ist nicht erforderlich.

Erforderliche Request Header:

```
Content-Type: application/xml
Accept: application/json
```

Ein Beispiel eines signierten claim XML-Dokumentes findet sich im Kap. 11.2.

9.2.2. Antwort

Im Erfolgsfall wird der HTTP Status-Code 200 zurückgeliefert und der Response Body ist leer. Im Fehlerfall wird eine JSON-Struktur gemäss Kapitel 9.6 zurückgegeben.

9.3. Methode *rt1-generate*

Vor dem Aufruf von *rt1-generate* muss die Urkundsperson die Prüfung der Berechtigung (siehe Kap. 7.3) durchlaufen.

Die Operation *rt1-generate* benötigt das ZB-Token, das PKCS#7-Objekt der Signatur, den damit signierten Hash, die Document-Revision der Signatur und die Domäne sowie den Kanton der Funktion, für welche eine Zulassungsbestätigung angebracht werden soll.

Diese Operation liefert das gerenderte, einzubettende Bild der Zulassungsbestätigung und zusätzliche Informationen zurück.

9.3.1. Aufruf

```
POST /CONTEXT-ROOT/zulab/rt1-generate
```

Die Attribute des JSON-Objektes im Request-Body sind wie folgt:

JSON Attribut	Format	Beschreibung
zb-token	String	Das zb-token (UUID) aus der Methode *startTransaction*.
pkcs7	String/PEM /PKCS7	Das PKCS#7 enthält das Signaturzertifikat, mit dem der untenstehende Hash signiert wurde. Das PKCS#7 ist im PEM Format. Es muss die gesamte Zertifikatskette vorhanden sein. Die Signatur muss einen qualifizierten Zeitstempel enthalten. Das PEM darf keine Steuerzeichen (z.B. Newlines) aufweisen.
hash	Objekt	
value	String	Der Hashwert des unsignierten Dokumentes (PDF) als Hex-kodierten String. Der Hash ist der Wert, mit welchem das PKCS#7 erstellt wurde. Dieser wird zur Überprüfung der PKCS#7-Datenstruktur benötigt.
algorithm	String	Bezeichnung des für die Berechnung des Hashwertes benutzten Algorithmus. Zulässige Werte sind: SHA-256, SHA-384, SHA-512, SHA3-256, SHA3-384, SHA3-512.
revision	Number	Revisionsnummer des PDF-Dokuments, welche die Signatur der Urkundsperson enthält. Die Zulassungsbestätigung wird nur für die angegebene Revision gültig sein.

Beispiel der JSON Struktur:

```
{
   "zb-token": "3d588980-bc3b-4e6d-b305-212eb2f844d9",
   "pkcs7": "9zCCBd+gAwIBAgIUJ1IRWxYc3k4B3iIAAG/JeIaLBxAwDQYJKoZIcNAQE/…/AG",
   "hash": {
     "value": "0E1D8F6E026349E5BB8BD98475429DABCF665C483129213C8BA6ECC24288231F",
     "algorithm": "SHA-256"
    },
   "revision": 2
}
```

Erforderliche Request Header:

```
Content-Type: application/json
Accept: application/json
```

9.3.2. Antwort

Im Erfolgsfall wird der HTTP Status-Code 200 zurückgeliefert und im Response-Body steht als Antwort ein JSON-Objekt. Im Request-Header ist der MIME-Type JSON angegeben.

```
Content-Type: application/json
```

Die Attribute des JSON-Objektes sind wie folgt.

JSON Attribut	Format	Beschreibung
signature-reason	String	String, welcher im PDF als Signaturgrund (signature reason) bei der Signatur hinterlegt werden muss.
cert-chain	String/PEM	Zertifikatskette der Server-Signatur (X.509 Zertifikate). Mit diesem Zertifikat wird die Zulassungsbestätigung signiert.
image	String/Base64/ PNG	Gerendertes, einzubettendes Bild der Zulassungsbestätigung im PNG-Format, Base64-kodiert.
layout	Objekt	Anweisungen für die Platzierung der Signatur.
left-pos	Number	Abstand vom linken Seitenrand in Pixel bei 72dpi. Positive ganze Zahl.
top-pos	Number	Abstand vom unteren Seitenrand in Pixel bei 72dpi. Positive ganze Zahl.
page	String	Seite, auf der die Zulassungsbestätigung angebracht werden muss. Zulässige Werte: FIRST: auf der ersten Seite PENULTIMATE: auf der vorletzten Seite ULTIMATE: auf der letzten Seite

Der Signaturgrund (*signature-reason*) ist ein Text mit Informationen, welcher als „Signature Reason“ beim Signieren des PDFs hinterlegt werden muss. Dieser String muss 1:1 übernommen werden.
Der String ist eine JSON-Datenstruktur, welche maschinell ausgewertet werden kann. Der Aufbau der JSON-Datenstruktur ist wie folgt:

JSON Attribut		**Beschreibung**
v		Die Version dieser Struktur (aktuell 2).
c		Seriennummer des Signaturzertifikates (aus PKCS#7-Struktur des RT1-Generate).
c3p		Optional. Für die Rückwärtskompatibilität mit der Vorgängerversion der REST-Schnittstelle. Issuer und Seriennummer des Authentication Certificates eines Drittanbieters.
t		Transaktions-UUID. Diese wird beim Logging für alle Einträge dieser Session verwendet.
h		SHA-256 Hash des generierten Bildes (siehe Wert von *image* in der Antwort des WebServices).
f		Liste der momentan gültigen Funktionen dieser Person im gewählten Kanton und der angegeben Domäne.
	fd	Domäne, in welcher die Funktion gültig ist, z.B. *not.*
	fi	Funktions-ID (Wert des liefernden Registers)
	fk	Kanton, in welchem die Funktion gültig ist, z.B. *BE.*
	fb	Volle Funktionsbeschreibung, z.B. *Notar / Notaire.*
	fo	UID der Organisation, an welche die Funktion gebunden ist.
	fp	Personen-ID der Person, welcher dieser Funktion zugeordnet ist (Wert des liefernden Registers).

Es ist zu beachten, dass die ID der Person und der Funktion diejenigen des anliefernden kantonalen Registers sind. Diese Identifikatoren sind nur innerhalb des Kantons und der Domäne eindeutig.

Beispiel der JSON-Struktur für die SignatureReason:

```
{
   "v":   2,
   "c":   "a38913a67b137b91",
   "t":   "41fd7eca-9bf6-48ff-82e6-eb049260f162",
   "h":   "f0f5db86868e794679c7251475bbf63ed029d2dcc49a1987b1f5ecb2df0f898d",
      "f": [
         {
            "fd": "upreg",
            "fi": "10001",
            "fk": "BE",
            "fb": "Notar/in - Notaire",
            "fo": "CHE-107450801"
            "fp": "1d32b4bc-b923-4615-9233-8bcbc5223a77"
         }
      ]
}
```

9.4. Methode *rt2-sign*

Diese Operation signiert den übergebenen Hash. Die zurückgegebene Signatur (PKCS#7) enthält Informationen (Transaktions-ID, Informationen über verwendetes Signaturzertifikat etc.), welche sich auf die Signatur der Urkundsperson beziehen. Diese Informationen werden von Signaturvalidatoren ausgewertet, um die Gültigkeit der Zulassungsbestätigung zu überprüfen. Weitere Informationen zu den eingebetteten CMS Signed Attributes finden sich in der Spezifikation der CMS Signed Attributes[2].

Die Methode ist nicht idempotent.

9.4.1. Aufruf

```
POST /CONTEXT-ROOT/zulab/rt2-sign
```

Es wird ein JSON-Objekt im HTTP Body übermittelt. Im Header muss der MIME-Type JSON angegeben werden.

```
Content-Type: application/json
```

Die Attribute des JSON-Objektes sind wie folgt:

JSON-Attribut	Format	Beschreibung
zb-token	String	Das zb-token (UUID) aus der Methode *startTransaction*. Nach Beendigung der Methode ist das Token «verbraucht» und kann nicht noch einmal verwendet werden.
revision	Number	Muss gleich dem Parameter *revision* + 1 in RT1-Generate sein.
hash	Objekt	
value	String	Hashwert der aktuellen PDF-Revision.
algorithm	String	Bezeichnung des für die Berechnung des Hashwertes benutzten Algorithmus. Zulässige Werte sind: SHA-256, SHA-384, SHA-512, SHA3-256, SHA3-384, SHA3-512

[2] CMS Attribute der Zulassungsbestätigung v3 – Spezifikation; elektronisch abrufbar unter unter www.bj.admin.ch > Wirtschaft > Elektronische öffentliche Urkunden und elektronische Beglaubigungen.

Beispiel der JSON-Struktur für den Aufruf:

```
{
   "zb-token": "3d588980-bc3b-4e6d-b305-212eb2f844d9",
   "revision": "3",
   "hash": {
     "value": "0E1D8F6E026349E5BB8BD98475429DABCF665C483129213C8BA6ECC24288231F",
     "algorithm": "SHA-256"
    }
}
```

Erforderliche Request Header:

```
Content-Type: application/json
Accept: application/json
```

9.4.2. Antwort

Im Erfolgsfall wird der HTTP Status-Code 200 zurückgeliefert und im Response-Body steht als Antwort ein JSON-Objekt. Im Request-Header ist der MIME-Type JSON angegeben.

```
Content-Type: application/json
```

Das Attribut des JSON-Objektes ist wie folgt:

JSON-Attribut	Format	Beschreibung
pkcs7	PKCS7/PEM	PKCS7 im PEM Format. Enthält die Serverzertifikate sowie die Signatur des Hashes aus dem Aufruf.

9.5. Methode *ping*

Damit externe Anbieter die Verfügbarkeit des Systems prüfen können, gibt es ein öffentliches Ping. Die Applikation macht hier sonst nichts.

9.5.1. Aufruf

```
GET /CONTEXT-ROOT/zulab/ping
```

9.5.2. Antwort

Nur der Status-Code ist relevant. Es wird kein Content zurückgeschickt. Falls der Service antwortet, wird mit dem HTTP Status-Code 200 geantwortet. Falls der Service zwar antwortet, aber aktuell Probleme bekannt sind, wird mit 503 geantwortet.

9.6. Antwort im Fehlerfall

Sofern einen Request keinen HTTP Status-Code 200 zurückliefert, wird eine JSON-Struktur zurückgeliefert, in welcher die Fehlerursache genauer beschrieben ist. Der in der Struktur zurückgegebene Fehlercode (error-code) kann vom Client ausgewertet werden. Der Fehlercode entspricht der Bedeutung des HTTP-Statuscodes der Antwort.

9.6.1. Format

Die JSON-Struktur in Fehlerfall hat die folgenden Attribute:

JSON Attribut	Format	Beschreibung
http-status	Integer	Entspricht dem HTTP-Statuscode der Antwort.
error-code	Integer	Fehlercode, wie in der folgenden Tabelle beschrieben.
description	String	Beschreibung des Fehlers. Diese wird aus der in der Applikationen geworfenen Ausnahmebedingung (exception) übernommen.
exception-class	String	Klassename der geworfenen Ausnahmebedingung.

9.6.2. Fehlercodes

Für diese Version der Schnittstelle sind folgende Fehlercodes definiert:

Fehler-code	Fehlerkategorie	Beschreibung	HTTP-Status
10	API	Ressource nicht gefunden.	404
11	API	HTTP-Methode wird für diesen Request nicht unterstützt.	405
12	API	Der in Content-Type gesetzte Medientyp wird von dieser Methode nicht unterstützt.	415
20	Format	Mindestens ein Parameter fehlt oder weist einen ungültigen Wert auf.	400
21	Format	Der Hash stimmt nicht mit dem Hash im PKCS#7 überein.	400
22	Format	Die PKCS#7-Struktur konnte nicht decodiert werden.	400
23	Format	Die Revisionsnummer, welche in RT1 angegeben wurde, wurde in RT2 nicht genau einmal inkrementiert.	400
24	Format	Der aktuelle Status der Transaktion erwartet den Aufruf einer anderen Methode.	400
25	Format	Der angegebene Provider ist nicht bekannt.	403
30	Session	Von dieser IP sind aktuell zu viele Transaktionen offen.	429
31	Session	Es konnte keine Transaktion für das gegebene Token gefunden werden. Gegebenenfalls ist die Session abgelaufen.	408
32	Session	Die Authentisierungs-Webseite wurde mit einem ungültigen/abgelaufenen Authentication-Session-Token aufgerufen.	408
40	Berechtigung	In derselben Domäne, im gleichen Kanton ist einer anderen Person dasselbe Zertifikat zugeordnet.	409
41	Berechtigung	Keine Person mit diesem Zertifikat gefunden.	403
42	Berechtigung	Diesem Zertifikat ist keine aktive Funktion zugeordnet.	403
43	Berechtigung	Das Signaturdatum ist vor dem Inkrafttreten der EÖBV für das entsprechende Register.	403
44	Berechtigung	Das Zertifikat der Signatur ist nicht der authentisierten Person zugeordnet.	403
99	Server	Es ist ein interner Server-Fehler aufgetreten.	500

9.6.3. Beispiel

Beispiel der JSON-Struktur einer Fehlermeldung:

```
Content-Type: application/json
```

```
{
    "http-status": 400,
    "error-code": 20,
    "description": "Number of transactions to start not specified.",
    "exception-class": "InvalidDataException"
}
```

10. Updateservice für Kantons- und Domänenliste

Um die Zulassungsbestätigung anzubringen, muss der Kanton und die Domäne angegeben werden, in welcher die Funktion der Urkundsperson registriert ist.

10.1. Update

Um nicht bei jedem Start der Client-Anwendung das gesamte XML herunterzuladen, antwortet der Webserver auf HTTP-HEAD Anfragen.

```
HEAD /CONTEXT-ROOT/zulab/list/update
```

So kann die Client-Anwendung gegen das gespeicherte letzte Änderungsdatum der Ressource prüfen und nur im Falle einer Änderung die ganze Liste herunterladen.

10.2. Aufruf

```
GET /CONTEXT-ROOT/zulab/list/update
```

10.2.1. Antwort

Als Antwort wird die Datei als *application/xml* mit den beiden Listen der Kantone und der Domänen zurückgeschickt. Diese UTF-8 kodierte XM-Datei entspricht folgender Definition:

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://www.glue.ch/localsigner/zulabconfiguration"
    xmlns="http://www.glue.ch/localsigner/zulabconfiguration" elementFormDefault="qualified">

    <xs:element name="config">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="comment" type="xs:string" minOccurs="0"/>
                <xs:element name="cantons" type="cantonsType"/>
                <xs:element name="domains" type="domainsType"/>
            </xs:sequence>
            <xs:attribute name="version" type="xs:dateTime"/>
        </xs:complexType>
    </xs:element>

    <xs:complexType name="cantonsType">
        <xs:sequence>
            <xs:element name="canton" type="entryType" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="domainsType">
        <xs:sequence>
            <xs:element name="domain" type="entryType" minOccurs="0" maxOccurs="unbounded"/>
```

```
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="entryType">
        <xs:sequence>
            <xs:element name="value" type="xs:string"/>
            <xs:element name="translations" type="i18nType"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="i18nType">
        <xs:sequence>
            <xs:element name="german" type="xs:string"/>
            <xs:element name="french" type="xs:string"/>
            <xs:element name="italian" type="xs:string"/>
        </xs:sequence>
    </xs:complexType>
</xs:schema>
```

11. Anhang

11.1. XML-Schema für claim

```
<?xml version="1.0" encoding="UTF-8"?>

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:claim="http://www.upreg.ch/claim/1"
    targetNamespace="http://www.upreg.ch/claim/1" elementFormDefault="qualified"
    xmlns:ds="http://www.w3.org/2000/09/xmldsig#">

    <xs:import namespace="http://www.w3.org/2000/09/xmldsig#"
        schemaLocation="https://www.w3.org/TR/2002/REC-xmldsig-core-20020212/xmldsig-core-schema.xsd"/>

    <xs:element name="claim" type="claim:claimType">
        <xs:annotation>
            <xs:documentation xml:lang="de"> Root Element für den Antrag an UPReg für die
                Authentisierung. </xs:documentation>
        </xs:annotation>
    </xs:element>

    <xs:complexType name="authTokenType">
        <xs:sequence>
            <xs:element name="authToken" type="claim:uuidType" minOccurs="1" maxOccurs="unbounded">
                <xs:annotation>
                    <xs:documentation xml:lang="de"> Der Authentisierungstoken für ein einzelnes
                        Dokument. </xs:documentation>
                </xs:annotation>
            </xs:element>
        </xs:sequence>
    </xs:complexType>

    <xs:simpleType name="uuidType">
        <xs:annotation>
            <xs:documentation xml:lang="de"> Datentyp für den Authentisierungstoken. Der Token ist
                eine UUID Type-4. </xs:documentation>
        </xs:annotation>
        <xs:restriction base="xs:string">
            <xs:length value="36"/>
        </xs:restriction>
    </xs:simpleType>

    <xs:simpleType name="cantonAbbreviationType">
        <xs:annotation>
            <xs:documentation xml:lang="de"> Datentyp für den Kantonskürzel nach eCH-0007.
                Zusätzlichen Kürzel: "CH" (die Zuständigkeit des Bundes betreffend) und "TS" (für
                Testzwecke) </xs:documentation>
        </xs:annotation>
        <xs:restriction base="xs:token">
            <xs:maxLength value="2"/>
            <xs:enumeration value="ZH"/>
```

```
            <xs:enumeration value="BE"/>
            <xs:enumeration value="LU"/>
            <xs:enumeration value="UR"/>
            <xs:enumeration value="SZ"/>
            <xs:enumeration value="OW"/>
            <xs:enumeration value="NW"/>
            <xs:enumeration value="GL"/>
            <xs:enumeration value="ZG"/>
            <xs:enumeration value="FR"/>
            <xs:enumeration value="SO"/>
            <xs:enumeration value="BS"/>
            <xs:enumeration value="BL"/>
            <xs:enumeration value="SH"/>
            <xs:enumeration value="AR"/>
            <xs:enumeration value="AI"/>
            <xs:enumeration value="SG"/>
            <xs:enumeration value="GR"/>
            <xs:enumeration value="AG"/>
            <xs:enumeration value="TG"/>
            <xs:enumeration value="TI"/>
            <xs:enumeration value="VD"/>
            <xs:enumeration value="VS"/>
            <xs:enumeration value="NE"/>
            <xs:enumeration value="GE"/>
            <xs:enumeration value="JU"/>
            <xs:enumeration value="CH"/>
            <xs:enumeration value="TS"/>
        </xs:restriction>
    </xs:simpleType>

    <xs:complexType name="claimType">
        <xs:sequence>
            <xs:element name="domain" type="xs:string">
                <xs:annotation>
                    <xs:documentation xml:lang="de"> Domäne, die die Zulassungsbestätigung eingeholt
                        werden soll. </xs:documentation>
                </xs:annotation>
            </xs:element>
            <xs:element name="canton" type="claim:cantonAbbreviationType">
                <xs:annotation>
                    <xs:documentation xml:lang="de"> Kanton, für welchen die Zulassungsbestätigung
                        eingeholt werden soll. </xs:documentation>
                </xs:annotation>
            </xs:element>
            <xs:element name="authTokens" type="claim:authTokenType">
                <xs:annotation>
                    <xs:documentation xml:lang="de"> Liste der Authentisierungstoken. Ein Token für
                        jedes Dokument, für das eine Zulassungsbestätigung angefordert wird.
                    </xs:documentation>
                </xs:annotation>
            </xs:element>
            <xs:element ref="ds:Signature">
                <xs:annotation>
                    <xs:documentation xml:lang="de"> Signatur der Urkundsperson mit qualifiziertem
                        Zertifikat. Ein Zeitstempel ist nicht erforderlich.
                        Canonicalization-Methode:
                        http://www.w3.org/TR/2001/REC-xml-c14n-20010315#WithComments (c14n mit
                        Kommentar). Signaturalgorithmus: RSA mit SHA256. Mit KeyInfo.
                    </xs:documentation>
                </xs:annotation>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
</xs:schema>
```

11.2. Beispiel für ein claim XML-Dokument

```
<?xml version="1.0"?>
<claim xmlns="http://www.upreg.ch/claim/1" xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.upreg.ch/claim/1
file:/G:/kunden/Bundesamt_fuer_Justiz/Notarenregister/11_UPReg_ohne_SuisseID/Zulab_REST_Schnitt-
stelle/XML%20Schemata/1.0/claim-1-0.xsd">
    <domain>upreg</domain>
    <canton>SO</canton>
    <authTokens>
        <authToken>840ace0b-2a6f-4657-9436-81820df6fa4f</authToken>
        <authToken>927e75f1-83c5-4424-9368-b2e9ef783e4d</authToken>
    </authTokens>
<ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
<ds:SignedInfo>
<ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#WithComments"/>
<ds:SignatureMethod Algorithm="http://www.w3.org/2001/04/xmldsig-more#rsa-sha256"/>
<ds:Reference URI="">
<ds:Transforms>
<ds:Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
<ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#WithComments"/>
</ds:Transforms>
<ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
<ds:DigestValue>wmMzDL7qem3wtfRuEbHU+7pF3ymfKGIiES7RDM0m3Do=</ds:DigestValue>
</ds:Reference>
</ds:SignedInfo>
<ds:SignatureValue>
MS21P9iDCAQgauNsOm9ozl/5BhmFFTMtcnYdBEtZOF0+pTYiFlp2W7uZJJ9q0R2rxmhhxz7VTVGi
PXsfBIKDAIyt63q2EkjJGoL8/kkknYnNhqIiSa0pHQ+HgKMeMJUuwCZsZZ2j6pvVBwemH+XVPNPU
CRUVlG31LeDMZHzhe4pslMjpdTAid8S73PTb9Bot2Gh1kEBcI8xoir6mCi5R3eQeLak3EyuHsRLJ
urkODaHP0U3oXNK320yYS1/OS4ZWzxJtHEel57mI4k0lp7yyqe+Wn68k4pqf5p4B4CxId6q+l/q4
iHj45r/eMxCUHfWVUHC2FuUm6QDonFLtNMzxdg==
</ds:SignatureValue>
<ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
<ds:X509Data>
<ds:X509SubjectName>1.2.840.113549.1.9.1=#160c7465737440676c75652e6368,CN=Test U-
ser,L=Bern,C=CH</ds:X509SubjectName>
<ds:X509Certificate>
MIIDLzCCAhegAwIBAgIBATANBgkqhkiG9w0BAQsFADBNMQswCQYDVQQGEwJDSDENMAsGA1UEBxME
QmVybjESMBAGA1UEAxMJVGVzdCBVc2VyMRswGQYJKoZIhvcNAQkBFgx0ZXN0QGdsdWUuY2gwHhcN
MjEwMTEyMTIwMjAwWhcNMzEwMTEyMTIwMjAwWjBNMQswCQYDVQQGEwJDSDENMAsGA1UEBxMEQmVy
bjESMBAGA1UEAxMJVGVzdCBVc2VyMRswGQYJKoZIhvcNAQkBFgx0ZXN0QGdsdWUuY2gwggEiMA0G
CSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQDbPSI82aNiL9rpd4BlUTf3eNQQTyoS6o/l299iVPEq
ymEZZszu5/APd8N/nYJ3iq72h/wZJIsFk7PIXSNvWBAzwlVdvtw85KS3grzht/mQcFiRb6gAF/T+
4eixmlyPGvp4IQIEP7tp6G9Ct9KLxBQjm8M5sBqG5j8ntMgj56XnQKM1gd6wEOxMGqWG3O3KZ+n2
IBBrvMVpO04jH/bipLOIF6i7UqZQNKzX5DbN9/bnZG6u81pkJ+6F2aBaYewhtHxQe416cXYmh3G/
aTpG4ybDKNUouPDu8eaj/nYxdoGPDnvwyUjMItPydYEw1ZsI6wfcRWdAphBoGfLfrBNOCLCVAgMB
AAGjGjAYMAkGA1UdEwQCMAAwCwYDVR0PBAQDAgTQMA0GCSqGSIb3DQEBCwUAA4IBAQDDmCIB7qNC
uYFJZ2+nKycXvEjneKh3Yu4A5L3GcWnX4ADW5Nx+y9QNY5vNIuDi8GwO5t3tlo4zb71/Qv4QPErw
ScAB3Um0Jh7/aacA3L7ffqACkvVwZ0hL+DTu7xMyJRtaE+KP6jvNFql29R0BQ5jfL9jkyWVBC5eI
ok5vjqxa8/B5WHllcKoxxrza4mJgRdDJ2iZor7xByEnV1mMdfCikXwkgTEn3FPoKwMtQnwDsWzbs
5NXzg+fKLrlvAPPmJ0+oJv8IrPkPh0pU9gDZUvo8DcotmHP8X9hIv5t7AKJEZH5D3VkFAAv1Kloa
/Dv0emb5t5hqW0w8WMmyJfmGVtoT
</ds:X509Certificate>
</ds:X509Data>
<ds:KeyValue>
<ds:RSAKeyValue>
<ds:Modulus>2z0iPNmjYi/a6XeAZVE393jUEE8qEuqP5dvfYlTxKsphGWbM7ufwD3fDf52Cd4qu9of8GSSLBZOz
yF0jb1gQM8JVXb7cPOSkt4K84bf5kHBYkW+oABf0/uHosZpcjxr6eCECBD+7aehvQrfSi8QUI5vD
ObAahuY/J7TII+el50CjNYHesBDsTBqlhtztymfp9iAQa7zFaTtOIx/24qSziBeou1KmUDSs1+Q2
zff252RurvNaZCfuhdmgWmHsIbR8UHuNenF2Jodxv2k6RuMmwyjVKLjw7vHmo/52MXaBjw578MlI
zCLT8nWBMNWbCOsH3EVnQKYQaBny36wTTgiwlQ==</ds:Modulus>
<ds:Exponent>AQAB</ds:Exponent>
</ds:RSAKeyValue>
</ds:KeyValue>
</ds:KeyInfo>
</ds:Signature></claim>
```