ZSR Webservice FAQ

Inhalt

Links

ApiGateway/swagger/index.html

ApiGateway/swagger/1/swagger.json

Swagger Editor - editor.swagger.io

OpenID Connect - openid.net

OAuth - oauth.net

JSON Web Tokens - jwt.io

ReDoc - redocly.github.io

W3 Status Code Definition - w3.org

Service Level Agreement IT Services SASIS

Abo-Optionen

Release Notes

Kontakt

Mapping Abo Files - Webservice

BusinessActivityEnumListe

Integration des Webservices

Welche Daten können über den ZSR-Webservice bezogen werden?

Die über den Webservice gelieferten Daten entsprechen grundsätzlich den **Daten die in den RK-Abos geliefert werden**. Welche Daten abgefragt werden können ist also abhängig von den Modulen die der Kunde abonniert hat. Siehe Abo-Optionen

Daten können abgefragt werden zu ZSR-Nummern / K-Nummern, bei welchen die Sistierung nicht länger als 10 Jahre zurückliegt.

Werden die Daten in gleicher Struktur geliefert wie in den Abo-Files?

Mit dem Webservice hat sich das Datenmodell (im Vergleich zu den Abo-Files) geändert. Siehe auch Mapping Abo Files - Webservice.

Die wichtigsten Entitäten sind

- **CareProvider** - Leistungserbringer
- **CareProviderBusiness** - Standort
- **ClearingNumber** - ZSR-Nummer
- **EmployeeNumber** - K-Nummer

clearingNumber (Zahlstelle)

- **number (ZSR-Nummer)**
- **correspondenceParty (Korrespondenzadresse)**
- **account (Konto)**
- **clearingNumberLaws (Zulassung)**
- **careProvider (Leistungserbringer)**
  - careProviderParties (Adressen)
  - qualifications (Qualifikationen)
  - employeeNumbers (K-Nummer)
- **careProviderBusinesses (Zahlstellen-Standorte)**
  - careProviderBusinessParties (Adressen)
  - facilities (Einrichtung)
  - healthServices (Methoden)
  - affiliations (Mitgliedschaften)
- **businessScope (OG/UG)**
- **businessActivity (Haupttätigkeit Ärztin/Arzt)**
- **relatedClearingNumbers (zugeordnete Zahlstellen/Zusätzliche Zahlstellen)**
- **relatedEmployees (Arbeitnehmer/Angestellte)**
- **licenses (Kantonale Bewilligung)**
- **admissions (Kantonale Zulassung)**
- **tariffSystems (Tarif-Verträge)**

> *Hinweise:*
> *Es handelt sich um eine stark vereinfachte Darstellung - für Details siehe Swagger-Schema.*
> *Berücksichtigt sind die Anpassungen gemäss Release Notes ZSR Webservice (24.06.22)*

Weshalb wurde das Datenmodell (im Vergleich zu den Abo-Files) verändert?

Gründe für die Änderung des Datenmodells:

- **Eine Wahrheit:** Das alte Datenmodell hat es zugelassen, dass für den gleichen LERB unterschiedliche Stammdaten geführt werden mussten. Dies gibt Unschärfen bei den fachlichen Daten und erschwert eine einheitliche Identifikation. Beim neuen Datenmodell wurde darauf geachtet, dass die Legacy-Daten über Zeit zusammengeführt werden können und pro LERB ein einheitlicher Datensatz geliefert wird. Der englische Arbeitsname der Applikation lautet denn auch CareProviderRegister (=Leistungserbringer-Register) und verwendet "Zahlstellenregister" nicht mehr.
- **Fachliche Daten:** Das alte Datenmodell konnte nicht in allen Bereichen eine fachliche Historie anlegen. Das neue Datenmodell sorgt dafür, dass eine fachliche Historisierung aller relevanten Daten möglich ist bzw. leicht ergänzt werden kann.
- **Zukunftssicherer:** Das alte Datenmodel konnte künftige Anforderungen ans Leistungserbringer-Register nur schwer abbilden. Mit dem neuen Datenmodell wurde Flexibilität geschaffen, um Änderungen der fachlichen Workflows schneller abbilden zu können.

Mit welchem Response-Format gibt der ZSR-Webservice Antwort?

Die Daten werden in **JSON** Format geliefert (XML wird nicht unterstützt).

Können Referenzdaten (=Stammdaten) separat geladen werden?

Stammdaten/Basisdaten zu ZSR-/K-Nummern Werten (Methoden, Qualifikationen, Banken, usf.) müssen grundsätzlich nicht separat geladen werden.
Alle notwendigen Daten werden direkt in der Detail-Response zur einzelnen Nummer mitgeliefert.

Bei Bedarf können Stammdaten/Basisdaten dennoch separat wie folgt eingelesen werden:

- Alle Enum-Werte (z.B. Gender, Canton u.a.) werden bereits im swagger.json (JSON) vollständig ausgewiesen. Mit Hilfe des SwaggerEditors oder anderer Tools lassen sich Referenz-Klassen des ZSR-Webservice samt der dazugehörigen Enum-Werte automatisch generieren.
- Nicht als Enum-Werte im swagger.json ausgewiesen, aber über einen dedizierten ZSR-Webservice Endpoint abrufbar sind folgende Werte:
  - Affiliations (Verbandscodes)
  - BusinessScopes (PAOG/PAUG)
  - ClearingNumberSuffixes (Nummernkreise)
  - Countries (Länder)
  - Facilities (Standorteigenschaften)
  - HealthServices (Methoden)

- Qualifications (Qualifikationen)
- TariffSystems (Tarifsysteme)

Die eindeutige Identifikation der Stammdaten ist über das Element "key" (string) möglich. Siehe Werden technische Id's geliefert?
Beispiel: "key": "3fa85f64-5717-4562-b3fc-2c963f66afa6"

Wo finde ich die technische Dokumentation des Webservice?

Es steht eine **Open API Specification** zur Verfügung.

Nebst Swagger kann beispielsweise ReDoc zur Darstellung OpenAPI Specification genutzt werden.

Gibt es eine Versionierung des ZSR-Webservice?

Der ZSR-Webservice liegt in der Version v1.x vor. Alle Änderungen am Webservice werden bis auf Weiteres nicht über neue Versionen im Parallelbetrieb umgesetzt, sondern alle Anpassungen erfolgen direkt auf der Version v1.x. Jegliche Änderungen werden jedoch rechtzeitig als Release-Notes publiziert und per Stage-System zur Verfügung gestellt. Es gelten die SASIS SLA.

Weil eine Versionierung des ZSR-Webservice bis auf Weiteres fehlen wird, empfiehlt die SASIS AG ihren Integratoren ein Adapterpattern zu verfolgen, das über eine releaseunabhängige Parametrierung eine Anpassung der Client-System Schnittstellen zum Webservice erlaubt.

Wie lauten die aktuellen IT-SLA der SASIS AG

Für den Webservice kommt das **Service Level Agreement** der SASIS IT Services zur Anwendung.

Anfragen und Support

Wo kann der Zugang zum ZSR-Webservice beantragt werden?

Bitte nehmen Sie Kontakt mit uns auf um den Zugang zum ZSR-Webservice zu beantragen.

An wen kann man sich bei Supportanfragen zum ZSR-Webservice wenden?

Bitte nehmen Sie Kontakt mit uns auf.

Authentifizierung und Absetzen von Abfragen

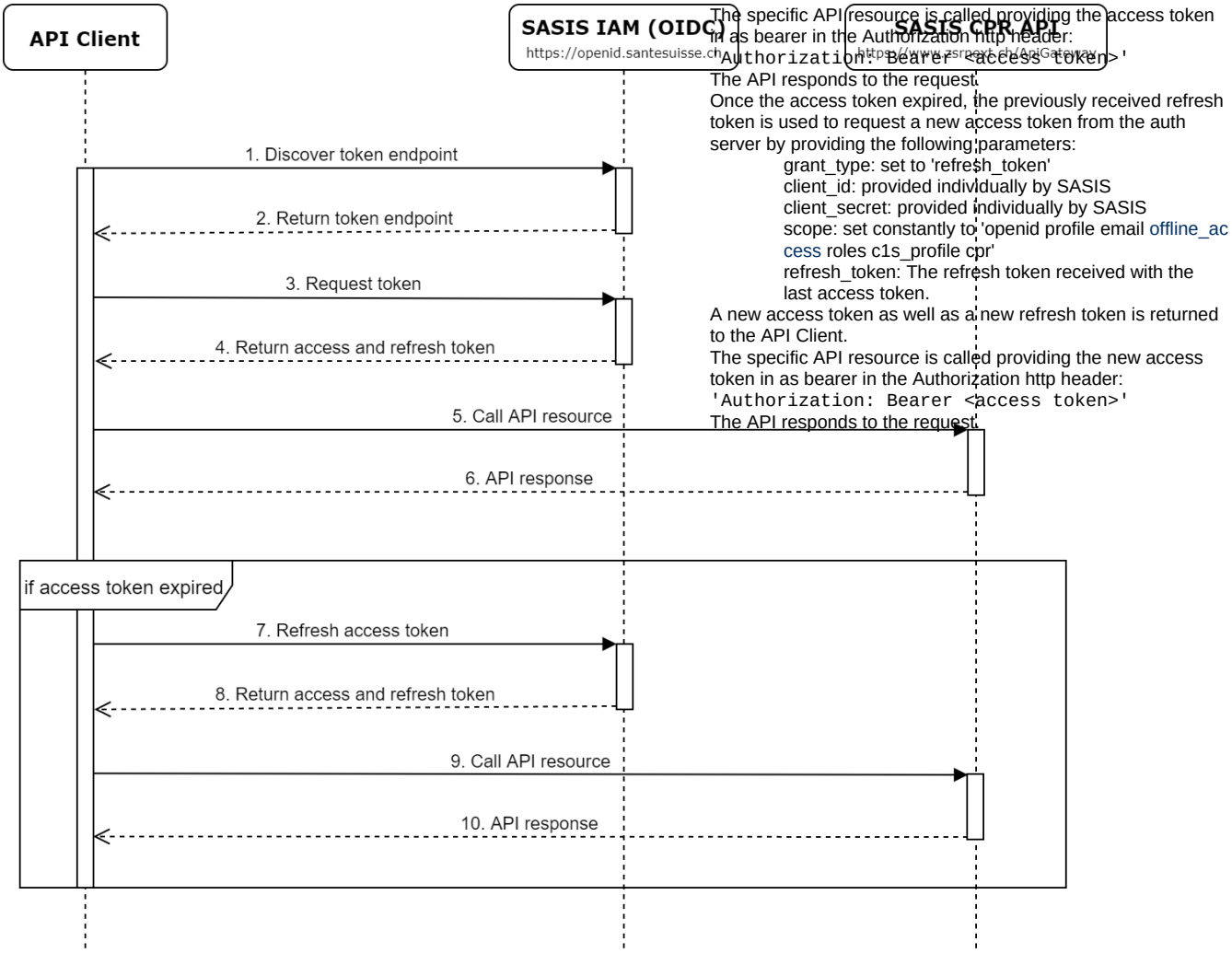
Wie kann auf den ZSR-Webservice zugegriffen werden?

Bitte nehmen Sie Kontakt mit uns auf um den Zugang zum ZSR-Webservice zu beantragen.

Wie erfolgt die Authentifizierung?

Die Authentifizierung basiert auf OpenID Connect bzw. OAuth 2.0 unter Verwendung von password grant.

1. The API Client requests the token endpoint from the auth server.
2. The token endpoint is returned to the API Client.
3. Request an access token from the auth server by providing the following post request parameters:
    a. grant_type: set to 'password'
    b. client_id: provided individually by SASIS
    c. client_secret: provided individually by SASIS
    d. username: provided individually by SASIS
    e. password: provided individually by SASIS
    f. scope: set constantly to 'openid profile email offline_access roles c1s_profile cpr'
4. The access token as well as the refresh token is returned to the API Client.



Access token

The access token response contains additional information:

Access token response

```
{
  "access_token": "MTQ0NjOkZmQ5OTM5NDE9ZTZjNGZmZjI3",
  "refresh_token": "GEbRxBNZmQOTM0NjOkZ5NDE9ZedjnXbL",
  "token_type": "bearer",
  "expires_in": 300,
  "scope": "openid profile email offline_access roles c1s_profile cpr"
}
```

The token itself is a JWT and can therefore be decoded on the JWT website.

The expires_in field defines the validity period of the token in seconds. Afterwards, a new token must be retrieved.

Code samples

A complete c# sample shows how to access one specific API resource (numbers):

CprApiAccessSample.zip

Authentication in other languages follows the same procedure.

The following code snippets explain the procedure on a step-by-step basis:

1./2. Retrieve the auth servers token endpoint

```
        /// <summary>
        /// Loads the IAM configuration.
        /// </summary>
        private async Task<DiscoveryResponse> GetDiscoveryDocumentAsync(CancellationToken ct)
        {
            using (var client = new HttpClient())
            {
                var discoveryResponse = await client.GetDiscoveryDocumentAsync(new DiscoveryDocumentRequest
                {
                    Address = "[AuthorityUrl]"
                }, ct).ConfigureAwait(false);

                if (discoveryResponse.IsError)
                    throw new Exception(discoveryResponse.Error);

                return discoveryResponse;
            }
        }
```

3./4. Request access and refresh token

```
        /// <summary>
        /// Gets a new token response with a username and password.
        /// </summary>
        private async Task<TokenResponse> RequestTokenAsync(CancellationToken ct)
        {
            using (var client = new HttpClient())
            {
                var discoveryResponse = await GetDiscoveryDocumentAsync(ct).ConfigureAwait(false);

                var tokenResponse = await client.RequestPasswordTokenAsync(new PasswordTokenRequest
                {
                    ClientId = "[ClientId]",
                    ClientSecret = "[ClientSecret]",
                    UserName = "[UserName]",
                    Password = "[Password]",
                    Address = discoveryResponse.TokenEndpoint,
                    GrantType = OidcConstants.GrantTypes.Password,
                    Scope = string.Join(" ", _scopes),
                }, ct).ConfigureAwait(false);

                if (tokenResponse.IsError)
                    throw new Exception(tokenResponse.Error);

                return tokenResponse;
            }
        }
```

7./8. Token refresh strategy based validity of cached token response and request new access token

```
        /// <summary>
        /// Gets the access token by either requesting a new token or by using the refresh token of an
already existing token.
        /// </summary>
        private async Task<string> GetAccessTokenAsync(CancellationToken ct)
        {
            if (_tokenResponse == null)
            {
                // Creates a new token response
                _tokenResponse = await RequestTokenAsync(ct).ConfigureAwait(false);
            }
            else
            {
                var jwtSecurityTokenHandler = new JwtSecurityTokenHandler();

                // Parses JWT access token
                var jwtSecurityToken = jwtSecurityTokenHandler.ReadToken(_tokenResponse.AccessToken) as
```

```
JwtSecurityToken;

                // The access token might be valid now, but expired the very next millisecond.
                // Thus, add a reasonable reserve in minutes for the validity time comparison below.
                var comparisionCorrectionInMinutes = 1;

                // Compares the access token life time with the current time, modified by the comparison
correction value.
                if (jwtSecurityToken.ValidTo < DateTime.UtcNow.AddMinutes(comparisionCorrectionInMinutes))
                {
                    // Updates the existing token response
                    _tokenResponse = await RefreshTokenAsync(_tokenResponse.RefreshToken, ct).ConfigureAwait
(false);
                }
            }

            return _tokenResponse.AccessToken;
        }

        /// <summary>
        /// Gets an updated token response by using a refresh token.
        /// </summary>
        private async Task<TokenResponse> RefreshTokenAsync(string refreshToken, CancellationToken ct)
        {
            using (var client = new HttpClient())
            {
                var discoveryResponse = await GetDiscoveryDocumentAsync(ct).ConfigureAwait(false);

                var tokenResponse = await client.RequestTokenAsync(new TokenRequest
                {
                    ClientId = "[ClientId]",
                    ClientSecret = "[ClientSecret]",
                    Address = discoveryResponse.TokenEndpoint,
                    ClientCredentialStyle = ClientCredentialStyle.AuthorizationHeader,
                    GrantType = OidcConstants.GrantTypes.RefreshToken,
                    Parameters =
                    {
                        { "refresh_token", refreshToken },
                        { "scope", string.Join(" ", _scopes) }
                    }
                });

                if (tokenResponse.IsError)
                    throw new Exception(tokenResponse.Error);

                return tokenResponse;
            }
        }
```

5./6./9./10. API resource call using the access token in the Authorization http header

```
        /// <summary>
        /// A simple CPR API number search request.
        /// </summary>
        private async Task<BulkResponse> CprApiSampleRequestAsync(string accessToken, CancellationToken ct)
        {
            BulkResponse bulkResponse = new BulkResponse();

            using (var client = new HttpClient())
            {
                client.SetBearerToken(accessToken);

                var response = await client.GetAsync($"https://[CprBaseUrl]/ApiGateway/api/v1/numbers?
searchOptions=Okp&offset=0&limit=10", ct).ConfigureAwait(false);

                if (!response.IsSuccessStatusCode)
                    throw new Exception("There was a problem with the request");
```

```
            string content = await response.Content.ReadAsStringAsync();

            if (content != null && content.Length > 0)
            {
                bulkResponse = JsonConvert.DeserializeObject<BulkResponse>(content);
            }
        }

        return bulkResponse;
    }
```

Plain API Call (putting everything together)

```
    var accessToken = await GetAccessTokenAsync(ct).ConfigureAwait(false);

    var cprApiResponse = await CprApiSampleRequestAsync(accessToken, ct).ConfigureAwait(false);
```

Welche Limiten bestehen auf Batchabfragen?

Als Batchabfragen gelten Abfragen mit mehr als **50 Requests pro Minute** und Kunde.

Abfragen aus Batch-Prozessen auf Online-Dienste der SASIS AG dürfen **ausschliesslich zwischen 22.00 und 4.00 Uhr** durchgeführt werden.
Die Online-Dienste der SASIS AG sind mit einer Begrenzung versehen, welche die Anfragen ab einer Überschreitung von **1000 Abfragen pro Minute** und Kunde zurückweist (http Statuscode 503).

Siehe auch Service Level Agreement der SASIS IT Services.

Weshalb kann ich nur 500 Nummern bei der Detailansicht beziehen?

Abfragen an die Endpoints ClearingNumbers und EmployeeNumbers sind aus Performance Gründen auf 500 Nummern limitiert. Der Abfrage-Prozess ist so zu planen, das jeweils maximal 500 Nummern in einem Request enthalten sind.

Welche Statuscodes werden vom Webservice zurückgegeben?

Der Webservice verwendet http status codes. Folgende Codes können zurückgegeben werden:

Status Code	Description
200	**Success**
202	**Accepted; Request is valid and business process could be triggered successfully.**
400	**Bad Request; The request data is invalid.**
401	**Unauthorized; The caller does not have sufficient privileges to perform the call.**
403	**Forbidden; The server is refusing the action.**
500	**Internal Server Error; Any unexpected internal failure.**

Wie können 503er-Fehler verhindert/vermieden werden?

Wird der Webservice mit zu vielen parallelen Abfragen überlastet, werden 503er-Fehler zurückgegeben. Um 503er-Fehler zu vermeiden, müssen die Anweisungen für das Laden der Nummernliste wie auch das Laden der Nummerndetails unbedingt beachtet werden (siehe auch Wie können Daten zu ZSR- und K-Nr. über den Webservice abgefragt werden?). Die Anzahl Requests kann weiter reduziert werden, indem beispielsweise nur Mutationen als Tagesscheiben abgefragt werden (siehe auch Wie können Mutationen abgefragt werden?). Nach dem Auftreten eines 503er-Fehlers sollte zudem ein Zeitpuffer von 5 Minuten bis zur nächsten Abfrage implementiert werden. Allenfalls ist auch zu prüfen, ob Webservice-Abfragen auf einen anderen, nächtlichen Zeitslot verlegt werden können

Wie können 400er-Fehler verhindert/vermieden werden?

Wenn der Client einen ungültigen Request an den Webservice sendet, wird ein 400er-Fehler zurückgegeben. Um 400er-Fehler zu vermeiden, müssen die Anweisungen für das Laden der Nummernliste wie auch das Laden der Nummerndetails unbedingt beachtet werden (siehe auch Wie können Daten zu ZSR- und K-Nr. über den Webservice abgefragt werden?). Der Abfrage-Prozess ist so zu planen, das jeweils maximal 500 Nummern in einem Request enthalten sind.

Abfrage von ZSR- und K-Nummern

Wie wird der Numbers Endpoint verwendet?

Der Endpoint Numbers liefert ZSR- und K-Nummern zurück die den Filterkriterien entsprechen.

Siehe auch Wie können Daten zu ZSR- und K-Nr. über den Webservice abgefragt werden?

Was sind die möglichen Filterkriterien beim Numbers Endpoint?

Filterkriterien:

- **filterOptions**: Begrenzt die Anzahl zurückgegebener Nummern auf bestimmte Kategorie(n)/Zertifizierer. Verfügbare Module analog der Abo-Optionen.
- **numberTypes**: Bestimmt den zurückgegebenen Nummerntyp (ZSR-/K-Nummer).
- **offset**: Paging-Wert, der eine Anzahl von Nummern überspringt. In der Regel kann hier 0 übergeben werden.
- **limit**: Paging-Wert, der die Anzahl zurückgegebener Nummern begrenzt. Damit mit einem Request alle Nummern auf einmal geladen werden können, darf hier ein relativ hoher Wert mitgegeben werden, z.B. 200'000.
- **modifiedFrom**: Das "Modified-Stichdatum", s. Abfrage von Mutationen (DIFF).

Gibt es eine erweiterte Suche?

Erweiterte Suchanfragen mit Filterkriterien auf Feldebene, beispielsweise nach Name oder Postleitzahl, werden über den ZSR-Webservice nicht angeboten. **Für erweiterte Suchanfragen steht die ZSR-Vollversion zur Verfügung.**

Für die Filterkriterien des Webservice siehe Datenabfrage.

Wie können Daten zu ZSR- und K-Nr. über den Webservice abgefragt werden?

Die Abfrage der Daten erfolgt in zwei Schritten.

Schritt 1 - Laden der Nummernliste:

Abfrage aller gewünschten ZSR-/K-Nummern über den Endpoint "Numbers". Angewendete Filterkriterien und die Benutzer-Berechtigungen beeinflussen das Suchresultat. Der Endpoint "Numbers" liefert sämtliche ZSR-/K-Nummern aus, die den Filterkriterien entsprechen und auf welche der Kunde berechtigt ist. Nicht geliefert werden ZSR-/K-Nummern die länger als 10 Jahre sistiert sind.

Filterkriterien:

- **filterOptions**: Begrenzt die Anzahl zurückgegebener Nummern auf bestimmte Kategorie(n)/Zertifizierer. Verfügbare Module analog der Abo-Optionen.
- **numberTypes**: Bestimmt den zurückgegebenen Nummerntyp (ZSR-/K-Nummer).
- **offset**: Paging-Wert, der eine Anzahl von Nummern überspringt. In der Regel kann hier 0 übergeben werden.
- **limit**: Paging-Wert, der die Anzahl zurückgegebener Nummern begrenzt. Damit mit einem Request alle Nummern auf einmal geladen werden können, darf hier ein relativ hoher Wert mitgegeben werden, z.B. 200'000.
- **modifiedFrom**: Das "Modified-Stichdatum", s. Abfrage von Mutationen (DIFF).

Schritt 2 - Laden der Nummerndetails:

Mit dem Resultat von Schritt 1 ist für jeweils 500 Nummern ein Request über den jeweiligen Detail-Endpoint abzusetzen. Die Response der Detail-Endpoints beinhaltet die kompletten Nummern-Details, auf die der User berechtigt ist. Auf gehäufte, parallele Einzelrequest ist aus performancegründen unbedingt zu verzichten.

Endpoints:

- für ZSR-Nummern: **Endpoint "ClearingNumbers"**
- für K-Nummern: **Eindpoint "EmployeeNumbers"**

Was sind Subscription Options?

Die über den Webservice gelieferten Daten entsprechen grundsätzlich den Daten die in den RK-Abos geliefert werden. **Die Subscription Options sind abhängig von den Modulen die der Kunde abonniert hat.** Siehe Abo-Optionen

Wie können die Subscription Options geändert werden?

Bitte nehmen Sie Kontakt mit uns auf.

Wie wird der ClearingNumbers Endpoint verwendet?

Der Endpoint ClearingNumbers liefert Details zu **ZSR-Nummern** zurück.

Siehe Wie können Daten zu ZSR- und K-Nr. über den Webservice abgefragt werden?

Wie wird der EmployeeNumbers Endpoint verwendet?

Der Endpoint EmployeeNumbers liefert Details zu **K-Nummern** zurück.

Siehe Wie können Daten zu ZSR- und K-Nr. über den Webservice abgefragt werden?

Wie können gelöschte Nummern identifiziert werden?

Wenn ein fachlicher Wert einmal geliefert wurde (mit einer bestimmten ID bzw. ZSR-/K-Nummer) und zu einem späteren Zeitpunkt **nicht mehr in der Response enthalten** ist, dann handelt es sich um einen gelöschten/stornierten Wert.

Wie können Mutationen abgefragt werden?

Über den Endpoint "Numbers" kann als Parameter ein **"Modified-Stichdatum"** festgelegt werden. In der Response enthalten sind dann nur Nummern, die seit diesem Datum modifiziert wurden. Solche Modifikationen können fachlicher Natur sein, aber auch nur rein technische Werte der Nummer betreffen.

Länger als 10 Jahre sistierte und stornierte Nummern werden über den Numbers-Endpoint nicht geliefert. Um herausfinden zu können, welche Nummern betroffen sind, müssen zuerst alle aktiven Nummern über den Numbers-Endpoint geladen werden. Fehlen Nummern in diesem Resultat, die im Client-System jedoch noch aktiv geführt werden, sind diese im Client-System zu beenden bzw. zu löschen.

Nach dem Laden aller modifizierten Nummern können die Details zu diesen Nummern wie üblich über die Detail-Endpoints abgefragt werden. Die Response dieser Endpoints liefert immer alle verfügbaren Daten zu einer ZSR-/K-Nummer und nicht nur die effektiv geänderten Werte.

Mutationen können vom Client-System z.B. **mittels Vergleich des JSON-Resultats der letzten Abfrage mit dem JSON-Resultat der neuen Abfrage** verarbeitet werden. Hierzu gäbe es folgende Fälle zu beachten:

- Element mit techn. ID x besteht im Client-System und wird im JSON-Resultat geliefert: Es ist zu prüfen, ob sich ein für das Client-System relevanter Feldwert des Elements geändert hat.
- Element mit techn. ID y besteht im Client-System und wird im JSON-Resultat nicht geliefert: Alle Feldwerte des Elements gelten als storniert und müssen entfernt werden.
- Element mit techn. ID z besteht nicht im Client-System und wird im JSON-Resultat geliefert: Alle Feldwerte des Elements sind neu und müssen eingelesen werden.

Die Integration des Webservice sollte immer so erfolgen, dass diese unabhängig von der Anzahl der gemeldeten Mutationen funktioniert, z.B. sollen selbst gewählte Thresholds nicht zum Abbruch des Daten-Imports führen.

Können auch nur geänderte ZSR-/K-Nummern geladen werden?

Ja, das Filterkriterium "modifiedFrom" erlaubt die Abfrage von Mutationen per Stichdatum. Siehe auch Wie können Mutationen abgefragt werden?

Wie sind Mutationen zu verarbeiten?

- **Element mit techn. ID x besteht im Client-System und wird im JSON-Resultat geliefert:** Es ist zu prüfen, ob sich ein für das Client-System relevanter Feldwert des Elements geändert hat.
- **Element mit techn. ID y besteht im Client-System und wird im JSON-Resultat nicht geliefert:** Alle Feldwerte des Elements gelten als storniert und müssen entfernt werden.
- **Element mit techn. ID z besteht nicht im Client-System und wird im JSON-Resultat geliefert:** Alle Feldwerte des Elements sind neu und müssen eingelesen werden.

Es gibt somit zwei Varianten wie eine Veränderung eines bestehenden Wertes geliefert werden kann:

- Variante 1: mit fachlicher Historisierung bei jeder Mutation des Feldwertes wird ein Element mit neuer technischer ID angelegt.
- Variante 2: noch ohne fachliche Historisierung ein Feldwert ändert sich, die technische ID bleibt gleich.

Im Endausbau des Webservice wird nur noch Variante 1 zur Anwendung kommen. Aktuell muss die Integration so erfolgen, dass die Client-Systeme mit beiden Varianten umgehen können.

Weshalb sind einzelne Einträge plötzlich nicht mehr in der Response enthalten?

Wurde ein Element bisher im JSON-Resultat geliefert und nun nicht mehr, dann handelt es sich um einen **Storno** (Falsche Erfassung von Daten bzw. nachträgliche Korrektur).

Wir empfehlen dieses Element im Client-System zu löschen.

Siehe auch Wie sind Mutationen zu verarbeiten?

Interpretation der Daten

Wie werden ZSR-Nummern Beziehungen im Webservice geliefert?

- **ClearingNumber.relatedClearingNumbers** enthält durch die SASIS manuell geführte Beziehungen zu ZSR-Nummern. Diese werden nur noch so lange geführt, bis vom Leistungserbringer bestätigte Daten vorliegen.
- **CareProvider.ClearingNumbers** enthält vom Leistungserbringer bestätigte Beziehungen zu ZSR-Nummern.
- **CareProvider.EmployeeNumbers** enthält vom Leistungserbringer bestätigte Beziehungen zu K-Nummern. (hierbei handelt es sich NICHT um Angestelltenverhältnisse).

Um die Gesamtheit der ZSR-Nummern Beziehungen zu erhalten, müssen somit sowohl **ClearingNumber.relatedClearingNumbers** wie auch **CareProvider.ClearingNumbers** berücksichtigt werden.

ClearingNumber.relatedClearingNumbers

```
    "relatedClearingNumbers": [
        {
          "clearingNumber": "B222222",
          "clearingNumberRelationType": "CHANGE_OF_HANDS",
          "clearingNumberRelationTypeTranslations": {
            "de": "Besitzerwechsel",
            "fr": "Changement de propriétaire",
            "it": "Cambiamento di proprietario"
          },
          "id": 520754
        }
      ]
```

CareProvider.ClearingNumbers

```
"clearingNumbers": [
          {
            "number": "N111111"
          }
        ]
```

CareProvider.EmployeeNumbers

```
 "employeeNumbers": [
          {
            "number": "999999K"
          }
        ]
```

Hinweis: Eine Beziehungsart wird nicht geführt und kann daher nicht ausgeliefert werden für **CareProvider.ClearingNumbers** und **CareProvider.EmployeeNumbers.**

Wie werden Angestelltenverhältnisse im Webservice geliefert?

Angestelltenverhältnisse werden unter clearingNumber.relatedEmployees geliefert.

Auf welchen Daten basiert die ausgewiesene Gültigkeit (validityPeriod) einer ZSR-/K-Nr?

Die Gültigkeit einer ZSR-/K-Nummer besteht nicht mehr nur aus einem Start- und Endedatum (Zeitraum). Die Gültigkeit wird anhand verschiedener Zeiträume bestimmt und als eine Liste von Gültigkeitsperioden (ValidityPeriods) ausgeliefert. Darin werden auch Gültigkeitslücken klar ausgewiesen.

Diese Gültigkeitsperioden werden wie folgt bestimmt:

ClearingNumber

- OKP: Start-Datum (S5) bis Ende-Datum (S6) plus Abzug von vergangenen bzw. aktiven Sistierungszeiträumen.
- OG52/56: Alle aggregierten Zeiträume der Methoden (HealthServices) plus gegebenenfalls Zeiträume von Standort-Eigenschaften (Facilities) wie FitnesClassification oder SPAK-Anerkennung.

EmployeeNumber

- Alle aggregierten Zeiträume der Anstellungsverhältnisse.

Wenn mehrere validityPeriods vorhanden sind (z.B. clearingNumber.validityPeriods) welche muss dann berücksichtigt werden?

Im Gegensatz zu den Abo-Files können über den Webservice mehrere Gültigkeitsbereiche (validityPeriods) zu einem Element ausgeliefert werden. **Gr undsätzlich sollten im Sinne einer fachlichen Historie alle vorhandenen Gültigkeitsperioden übernommen werden**. Ist dies nicht möglich, empfehlen wir die aktuellste bzw. letzte gültige zu berücksichtigen.

Wie kann ich feststellen, ob es sich um ein Bank- oder Postkonto handelt?

Die Klassifizierung Bankkonto/Postkonto wird im ZSR-Webservice nicht mehr geführt.

Ob ESR Zahlungen möglich sind, ist unter ClearingNumberAccount anhand vom Feld hasPaymentOrderReferenceNumber eruierbar.

Endpoint: ClearingNumbers, Objekt: clearingNumberAccount

Welche Bedeutung haben die Daten «0001-01-01» bzw. «9999-12-31»?

Für fachliche Zeiträume liefert der ZSR-Webservice immer einen Wert für Start- und Ende-Datum, auch dort wo sich aus den Legacy-Daten oder aus anderen Gründen sinnvollerweise keine genauen Datumswerte bestimmen lassen.

Dafür werden je einen Platzhalterwert für das Start- und das Ende-Datum verwendet. Die Interpretation dieser Werte ist wie folgt:

- **«0001-01-01»** ist ein **Platzhalterwert für ein Startdatum** und bedeutet, dass es kein bekanntes fachliches Anfangsdatum gibt (NULL).
- **«9999-12-31»** ist ein Platzhalterwert für ein Endedatum und bedeutet, dass es kein bekanntes fachliches Enddatum gibt (NULL).

Ein fachlicher Wert mit fachlichem Startdatum «0001-01-01» und fachlichem Endedatum «9999-12-31» ist somit zu jedem Zeitpunkt gültig.

Werden technische Id's geliefert?

Element id

Die in der Detail-Response gelieferten IDs haben keinerlei fachliche Bedeutung und sollten nicht zur fachlichen Interpretation von Werten herbeigezogen werden. Sie **dienen ausschliesslich der technischen Identifikation der gelieferten Daten** bzw. können zur Handhabung von Mutationen (siehe Wie können Mutationen abgefragt werden? bzw. Wie sind Mutationen zu verarbeiten?) genutzt werden. Auf der technischen IDs sollte nie eine Logik codiert werden.

Für den Hauptrecord der ZSR-/K-Nummer wird ausserdem überhaupt keine technische ID geliefert. Eine Identifikation ist nur mittels ZSR-/K-Nummer möglich. Ein Ersatzwert zur vormaligen ZahlstellenId wird nicht ausgeliefert

Element key

"Stammdaten/Referenzdaten", z.B. wie Kanton, Land, Qualifikation usf., werden mit einer beständigen GUID als Identifikator ausgeliefert. Siehe auch Können Referenzdaten (=Stammdaten) separat geladen werden?

Die eindeutige Identifikation ist über das Element "key" (string) möglich bei folgenden Entitäten:

- Affiliations
- BusinessScopes
- ClearingNumberSuffixes
- Countries
- Facilities
- HealthServices
- Qualifications
- TariffSystems

Beispiel: `"key": "3fa85f64-5717-4562-b3fc-2c963f66afa6"`

Wie können Dummy-Nummern identifiziert werden?

Die sogenannten Dummy-Nummern sind ein weiterhin verwendetes Legacy-Konstrukt, das weiterhin per ZSR-Webservice bezogen werden kann.

Sobald auf dem ClearingNumber Objekt ein Wert für das **Property "clearingNumberDummy"** geliefert wird, handelt es sich um eine Dummy-Nummer.

Beispiel-Struktur:

```
{
        "clearingNumber": {
                "clearingNumberDummy": {
                        "id": 912
                        "name": "CH-Arzt",
                }
        }
}
```

Wie werden Partnerart-Obergruppe/Partnerart-Untergruppe im Webservice ausgewiesen?

Alle Leistungserbringer ausser Ärzte und Ärztinnen

- Im Feld parentBusinessScope wird immer der Wert für die Partnerart-Obergruppe geliefert.
- Im Feld businessScope wird
  - falls vorhanden der Wert der Partnerart-Untergruppe geliefert.
  - ansonsten wird der Wert der Partnerart-Obergruppe geliefert.

Ärzte und Ärztinnen
"Ärzte und Ärztinnen" werden in businessScope und businessActivity kategorisiert.

Die Werte für BusinessScopes/ParentBusinessScopes können Sie via Webservice über den Endpoint «BusinessScopes» abfragen: /api/v1 /businessscopes.

Die Werte für businessActivity sind als Enum-Werte im swagger.json (JSON) ausgewiesen.

Siehe auch Mapping Abo Files - Webservice,Tabs PAOG_BusinesssScope, PAOG_BusinesssScope, PAUG_BusinessActivities

Allgemeine Informationen zum Zahlstellenregister

Wie setzt sich eine gültige ZSR-Nummer zusammen?

Die ZSR-Nummer setzt sich zusammen aus einem Prüfbuchstaben (1. Stelle), Laufnummer (Stellen 2-5) und dem Nummernkreis (Stellen 6-7).

Prüfbuchstabe 1
Laufnummer 4
Kantons-Identifikation 2
Beispiel: L248519
L = Prüfbuchstabe
2485 = Laufnummer
19 = Nummernkreis

Berechnung Prüfbuchstabe
Jeder Wert wird zuerst mit seiner Stellenposition multipliziert (letzte Zahl = Stellenposition 1).
Die Resultate werden anschliessend summiert.
Die Summe wird durch die Anzahl Buchstaben im Alphabet geteilt (Modulo 26).
Der Restbetrag ergibt die Stelle des Buchstabens im Alphabet

Beispiel: L248519

(9*1)+(1*2)+(5*3)+(8*4)+(4*5)+(2*6)=90
90/26=4.461...
90-(3*26)=12
12ter Buchstabe im Alphabet= L

Beispiel: Y274589

(9*1)+(8*2)+(5*3)+(4*4)+(7*5)+(2*6)=103
103/26=3.96...103-(26*3)=25
25ter Buchstabe im Alphabet = Y

Wie setzt sich eine gültige K-Nummer zusammen?

K-Nummern setzen sich zusammen aus einer Laufnummer (Stellen 1-6) und dem Buchstaben "K" (7.Stelle).