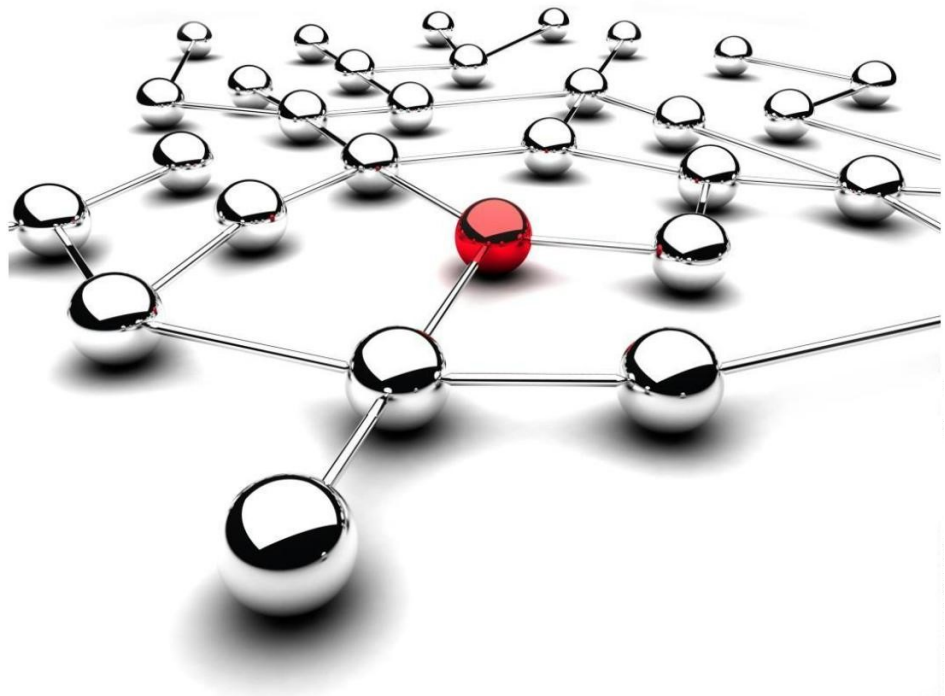




REST-API

Kurzanleitung und erste Schritte...



I. Allgemeines

Die Orgasoft.NET REST-API wird beim Anwender lokal (in der Regel in einer DMZ) installiert und stellt eine zentrale Schnittstelle für alle externen Anwendungen zur Verfügung, die Stamm- oder Bewegungsdaten mit dem System austauschen wollen.

Je nach Berechtigung kann man dabei u.a. auf folgende Daten zugreifen:

- Artikel mit zahlreichen Zusatzinformationen
- Adressen
- Waren- und Hauptgruppen
- Rezepturen
- Geschäftsvorfälle aller Art (z.B. Bestellungen, Lieferscheine, Rechnungen, etc.)
- Inventuren
- Adress-Artikel-Historie und Adress-Vorfall-Historie

Ferner können Kassensysteme die gebuchten Umsätze über die REST-API an Orgasoft.NET übermitteln.

II. URL

URL und Port der REST-API sind frei konfigurierbar, das Schema ist in der Regel folgendermaßen aufgebaut:

<https://<IP|Domain>:<Port>/api/v1/>

Ob und ggf. unter welcher URL eine Staging Umgebung für die Entwicklungsphase zur Verfügung steht, kann Ihnen der Administrator des Anwenders beantworten.

Eine Beispiel-Installation zum Testen des Zugriffs finden Sie unter

<https://api.orgasoft.net:8001/api/v1/>

Der vollständige Befehl, um sich am Beispiel-Service anzumelden lautet:

```
curl -X GET "https://api.orgasoft.net:8001/api/v1/authenticate" -H "accept: application/json" -H "authorization: Basic c3lzOmFkbWlu"
```

I. API-Dokumentation

Die Orgasoft.NET REST-API liefert (konfigurierbar) bei jeder Installation die aktuelle Dokumentation in der OpenAPI Spezifikation Ver. 2 (auch bekannt als Swagger-Dokumentation) mit.

Durch diese Art der Dokumentation kann man jederzeit interaktiv Calls mit den Original-Daten des Anwenders ausprobieren und bekommt die Ergebnisse angezeigt.

Die API-Dokumentation erreichen Sie unter folgender URL:

<https://<IP|Domain>:<Port>/api/v1/docs/>

Die Dokumentation der Beispiel-Installation des Service steht demnach unter der folgenden Adresse zur Verfügung:

<https://api.orgasoft.net:8001/api/v1/docs/>

Die gesamte Dokumentation lässt sich auch als json-Datei herunterladen und kann so beispielsweise verwendet werden, um sich automatisch einen entsprechenden Client erzeugen zu lassen (beispielsweise unter <https://editor.swagger.io/>). Die URL für diese Datei lautet:

[https:// <IP|Domain>:<Port>/api/v1/docs/swagger.json](https://<IP|Domain>:<Port>/api/v1/docs/swagger.json)

Für die Dokumentation des Beispiel-Services also

<https://api.orgasoft.net:8001/api/v1/docs/swagger.json>

II. Authentifizierung

Es gibt zwei grundsätzliche Methoden der Authentifizierung:

1. Eine Anmeldung mit Benutzername und Passwort. Hierfür wird die Methode der [basic http authorization](#) angewandt. In der Antwort erhält man ein AccessToken, das in allen nachfolgenden Calls als *http bearer token* oder als *X-API-KEY* verwendet werden kann. Dieser AccessToken hat i.d.R. nur eine begrenzte (konfigurierbare) Gültigkeit.
2. Die Verwendung eines [X-API-KEY](#). Der API-KEY stammt entweder aus der Anmeldung in 1) oder wird während der Konfiguration der REST-API fest vergeben. Die Gültigkeit des fest vergebenen API-KEY ist unbegrenzt. Der *X-API-KEY* wird im Header des Calls übergeben.

Beispiel:

```
curl -X GET "https://128.0.0.1:8000/api/v1/articlegroups" -H "accept: application/json" -H "X-API-KEY: A=kHzmA0vweZvV)"
```

III. Autorisierung

Die Zugriffsrechte des angemeldeten Benutzers bzw. die dem API-KEY zugeordneten Rechte werden über die Zugangsberechtigungen in Orgasoft.NET verwaltet (Rolle: Web).

IV. Return-Codes und Responses

Die Calls liefern über den *http response code* zurück, ob sie erfolgreich ausgeführt wurden oder nicht. Hierbei werden folgende Codes verwendet:

Code	Bedeutung	Erläuterung
200	OK	Anfrage wurde erfolgreich ausgeführt, Ergebnis ggf. im Response.
401	Unauthorized	Authentifizierung wurde noch nicht durchgeführt
403	Forbidden	Der API-User ist authentifiziert, darf aber diese Operation nicht durchführen. Die Berechtigung muss über Orgasoft.NET erteilt werden.
404	Not found	Die angefragte Ressource (z.B. Artikel, Adresse) wurde nicht gefunden.
500	ServerError	Beim Verarbeiten der Anfrage ist ein interner Fehler aufgetreten. Genauere Informationen sind dem Log von Orgasoft.NET zu entnehmen.

Im Falle von 200 OK werden i.d.R. die entsprechenden Daten im json-Format im Resonse enthalten sein. Zusätzlich enthält jede Antwort (auch im Fehlerfall) ein Result-Objekt, das weitere Informationen über das Ergebnis der Anfrage bereitstellt. Im Fehlerfall (z.B. bei der Neuanlage eines Objektes via POST) können hier wichtige Informationen zur Art des Fehlers oder der fehlerhaften Daten enthalten sein.

Beispiel-Antwort bei der Neuanlage einer Adresse:

```
{
  "Result": {
    "Code": 1,
    "Message": "Address with ID 7099 could not be created: Field 'Name1' is
required.",
    "Details": "ServerError"
  }
}
```

V. Schritt für Schritt-Anleitung

Rufen Sie die API-Dokumentation der Beispiel-Installation der REST-API unter folgender URL auf:

<https://api.orgasoft.net:8001/api/v1/docs/>

Klicken Sie auf den Button



Im Fenster, das sich nun öffnet, geben Sie als Username **sys** und als Passwort **admin** ein:

Available authorizations ✕

Basic authorization

Use basic authorization to authenticate a user. The returned access token of the GET authenticate call can subsequently be used as an API KEY or Http Bearer token.

Username:

2

Password:

3

4 Authorize Close

api-security (apiKey)

Authorizes the call with the access token returned from basic authorization or with a permanent API KEY (if configured).

Name: X-API-KEY

In: header

Value:

Authorize Close

Damit sind die Anmeldedaten für die nun folgende Authentifizierung hinterlegt. Schließen Sie das Fenster mit Close:

Available authorizations ✕

Basic authorization

Authorized

Use basic authorization to authenticate a user. The returned access token of the GET authenticate call can subsequently be used as an API KEY or Http Bearer token.

Username: `sys`
Password: `*****`

Logout Close 5

api-security (apiKey)

Authorizes the call with the access token returned from basic authorization or with a permanent API KEY (if configured).

Name: X-API-KEY
In: header
Value:

Authorize Close

Scrollen Sie nach unten bis zum Abschnitt **authentication**. Das geschlossene Schloss zeigt an, dass man die Funktion nun ausführen darf. Klicken Sie auf die Zeile mit dem **GET /authenticate**:

authentication ▼

POST	/authenticate	Authenticate a user and return access token	
GET	/authenticate	Authenticate a user and return access token	6 
POST	/logout	Log out current user	
GET	/token	Get Token	

Anschließend klicken Sie auf **Try it out**:

authentication 

POST `/authenticate` Authenticate a user and return access token

GET `/authenticate` Authenticate a user and return access token 

Authenticate a user using http Basic authorization scheme and return an access token that can be used for all subsequent calls with http Bearer authorization.

Parameters 7 Try it out

No parameters

Responses Response content type application/json 

... und dann auf **Execute**:

GET `/authenticate` Authenticate a user and return access token 

Authenticate a user using http Basic authorization scheme and return an access token that can be used for all subsequent calls with http Bearer authorization.

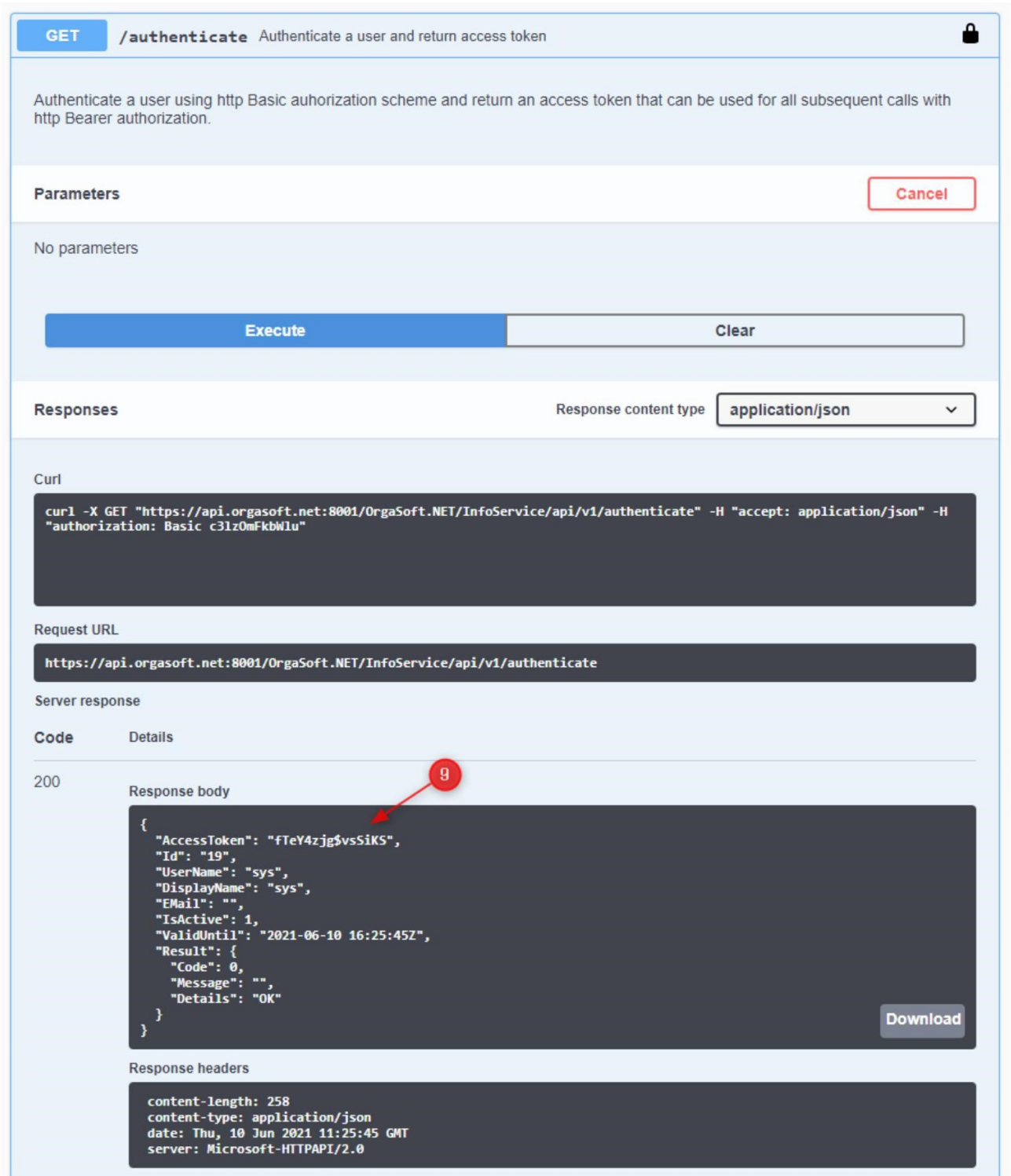
Parameters Cancel

No parameters

8 Execute

Responses Response content type application/json 

Als Ergebnis erhalten Sie ein **http-Code 200** und folgenden Response im json-Format:



The screenshot displays the API testing interface for the endpoint `/authenticate`. The method is `GET`. The description states: "Authenticate a user using http Basic auhorization scheme and return an access token that can be used for all subsequent calls with http Bearer authorization." The parameters section is empty. The response content type is set to `application/json`. The curl command is: `curl -X GET "https://api.orgasoft.net:8001/OrgaSoft.NET/InfoService/api/v1/authenticate" -H "accept: application/json" -H "authorization: Basic c3lzOmFkbWlu"`. The request URL is `https://api.orgasoft.net:8001/OrgaSoft.NET/InfoService/api/v1/authenticate`. The server response is a `200` status code. The response body is a JSON object containing an `AccessToken` and user details. A red circle with the number `9` highlights the `AccessToken` value. The response headers include `content-length: 258`, `content-type: application/json`, `date: Thu, 10 Jun 2021 11:25:45 GMT`, and `server: Microsoft-HTTPAPI/2.0`.

GET `/authenticate` Authenticate a user and return access token

Authenticate a user using http Basic auhorization scheme and return an access token that can be used for all subsequent calls with http Bearer authorization.

Parameters Cancel

No parameters

Execute **Clear**

Responses Response content type `application/json`

Curl

```
curl -X GET "https://api.orgasoft.net:8001/OrgaSoft.NET/InfoService/api/v1/authenticate" -H "accept: application/json" -H "authorization: Basic c3lzOmFkbWlu"
```

Request URL

```
https://api.orgasoft.net:8001/OrgaSoft.NET/InfoService/api/v1/authenticate
```

Server response

Code **Details**

200

Response body

```
{
  "AccessToken": "fTeY4zjg$vsSiKS",
  "Id": "19",
  "UserName": "sys",
  "DisplayName": "sys",
  "EMail": "",
  "IsActive": 1,
  "ValidUntil": "2021-06-10 16:25:45Z",
  "Result": {
    "Code": 0,
    "Message": "",
    "Details": "OK"
  }
}
```

Response headers

```
content-length: 258
content-type: application/json
date: Thu, 10 Jun 2021 11:25:45 GMT
server: Microsoft-HTTPAPI/2.0
```

Download

Bitte kopieren Sie den AccessToken (9) ohne die Anführungszeichen.

Scrollen Sie wieder nach oben auf die Seite und wählen Sie erneut



Available authorizations x

Basic authorization

Authorized

Use basic authorization to authenticate a user. The returned access token of the GET authenticate call can subsequently be used as an API KEY or Http Bearer token.

Username: sys
Password: *****

Logout

Close

api-security (apiKey)

Authorizes the call with the access token returned from basic authorization or with a permanent API KEY (if configured).

Name: X-API-KEY
In: header
Value:

fTeY4zjg\$vsSiKS 11

12 **Authorize**

Close

Fügen Sie den AccessToken unter (11) ein und bestätigen Sie mit (12). Schließen Sie das Fenster mit Close.

Anschließend sollten alle Schloss-Symbole schwarz und geschlossen sein, was anzeigt, dass Sie nun alle Funktionen verwenden können:

address ▼

GET /addresses Get all addresses 

POST /addresses Create new address 

Ausprobieren können Sie alle Funktionen mit dem Button **Try it out**. Eventuell benötigte Parameter werden Ihnen angezeigt und können gefüllt werden. Unbedingt notwendige Parameter sind mit **required** gekennzeichnet:

GET `/addresses/{id}` Get address 

Returns the requested address.

Parameters

Cancel

Name	Description
id * required string (path)	10003

Execute

Clear

Responses

Response content type **application/json** ▼

Curl

```
curl -X GET "https://api.orgasoft.net:8001/OrgaSoft.NET/InfoService/api/v1/addresses/10003" -H "accept: application/json" -H "X-API-KEY: fTeY4zjg$vsSiKS"
```

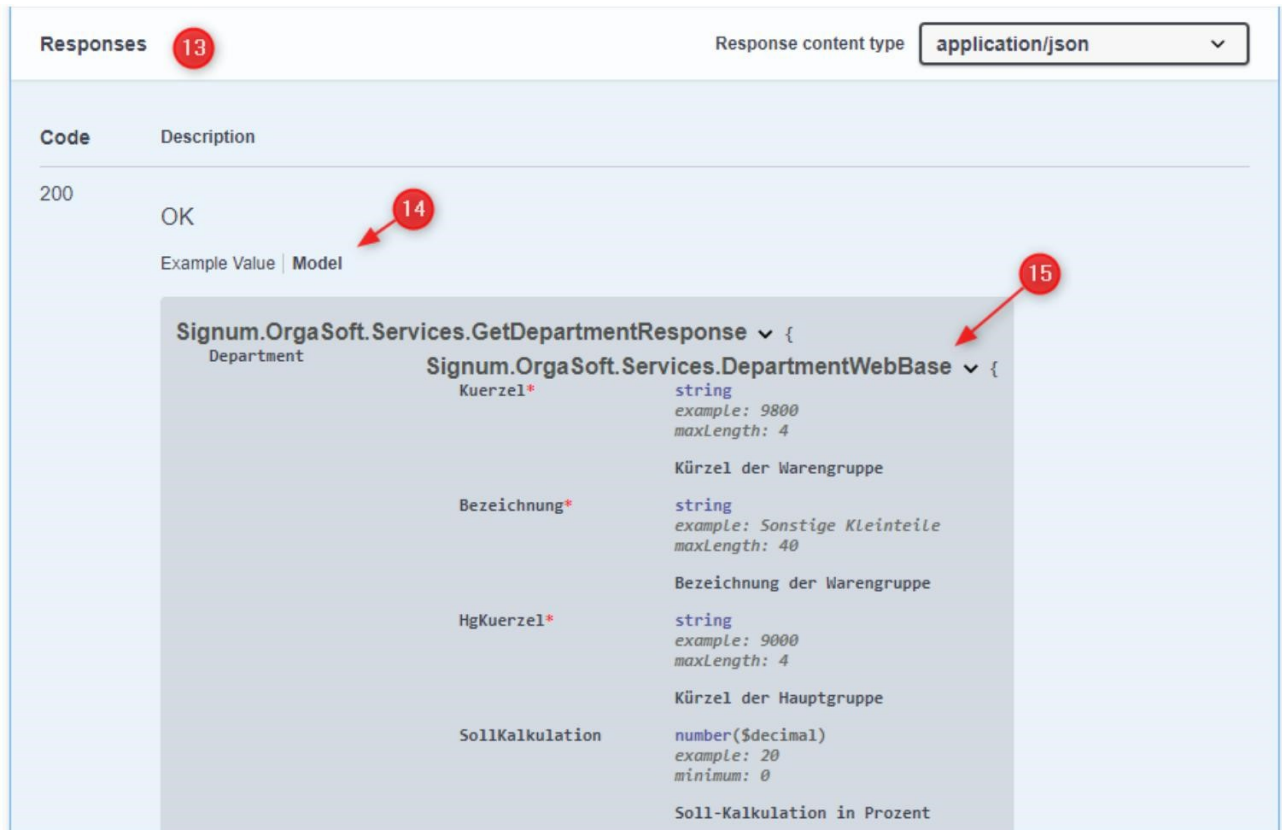
Request URL

```
https://api.orgasoft.net:8001/OrgaSoft.NET/InfoService/api/v1/addresses/10003
```

Server response

Code	Details
200	Response body <pre>{ "Address": { "KorrNr": "10003", "Suchname": "GOETHE-GYMNASIUM", "Kunde": 1, "Lieferant": 0, "Interessent": 0, "Filiale": 0,</pre>

Eine Beschreibung, welches Feld im Ergebnis was bedeutet, sehen Sie, wenn Sie unter **Responses** (13) auf **Model** (14) klicken und dann das Objekt (15) anklicken:



The screenshot displays the 'Responses' section of the Orgasoft.NET API documentation. The 'Response content type' is set to 'application/json'. The 'Code' column shows '200' and the 'Description' column shows 'OK'. The 'Example Value' column has a tab labeled 'Model' (14). The expanded JSON object (15) is as follows:

```
Signum.OrgaSoft.Services.GetDepartmentResponse {
  Department: Signum.OrgaSoft.Services.DepartmentWebBase {
    Kuerzel*: string
      example: 9800
      maxLength: 4
      Kürzel der Warengruppe
    Bezeichnung*: string
      example: Sonstige Kleinteile
      maxLength: 40
      Bezeichnung der Warengruppe
    HgKuerzel*: string
      example: 9000
      maxLength: 4
      Kürzel der Hauptgruppe
    SollKalkulation: number($decimal)
      example: 20
      minimum: 0
      Soll-Kalkulation in Prozent
  }
}
```

VI. ListRequest

Der ListRequest kann bei allen Search-POSTs im Body übergeben werden und spezifiziert genauer, welche Daten zurückgeliefert werden sollen. Das Model des ListRequest ist nicht ganz selbsterklärend, daher an dieser Stelle noch ein paar ergänzende Erläuterungen:

EqualityFilter:

Mit dem EqualityFilter können ein oder mehrere Bedingungen übergeben werden, die das gesuchte Objekt erfüllen muss. Wenn mehrere EqualityFilter übergeben werden, müssen alle erfüllt werden, damit das Objekt zurückgeliefert wird.

Folgendes wäre z.B. ein gültiger EqualityFilter für die Suche:

```
{
  "Take": "20",
  "EqualityFilter": [
    {
      "Key": "HauptLieferant",
      "Value": "70001"
    },
    {
      "Key": "WgKuerzel",
      "Value": "1502"
    }
  ]
}
```

Criteria:

Mit dem Criteria-Objekt können komplexere, verkettete Abfrageausdrücke für die Selektion der zu liefernden Daten definiert werden. Das Criteria-Objekt hat folgende Syntax:

Criteria: <BinaryCriteria | UnaryCriteria | LogicalCriteria >

Beispiel 1: [{"Vk1"}, ">=", 10]

Beispiel 2: [{"Vk1"}, ">=", 10], "AND", [{"Vk3"}, ">", 0]

Beispiel 3: [{"WgKuerzel"}, "=", "LBM2"], "OR", [{"WgKuerzel"}, "=", "TABAK"]]

Beispiel 4: [{"Vk1"}, ">=", 10], "AND", [{"Kurztext"}, "like", "%schale%"]]

Beispiel 5: [{"ArtikelNr"}, "in", [{"4711", "4712", "4713", "4714"}]]]

BinaryCriteria: [Field, Operator, Value]

Beispiel 1: [{"Vk1"}, ">=", 10]

UnaryCriteria: [UnaryOperator, Field]

Beispiel 1: ["is null", ["HauptLieferant"]]

LogicalCriteria: [Criteria, LogicalOperator, Criteria]

Beispiel 1: [{"Vk1"}, ">=", 10], "AND", ["is null", ["HauptLieferant"]]]]

Field: [PropertyName]

Beispiel 1: ["WgKuerzel"]

Beispiel 2: ["Vk1"]

Value: < String | Number | [Array] >

Beispiel 1: "Name"

Beispiel 2: 1.5

Beispiel 3: [{"4711", "4712", "4713", "4714"}]]]

Operator: < "=" | "<=" | ">=" | ">" | "<" | "!=" | "like" | "not like" | "in" | "not in" >

LogicalOperator: < "AND" | "OR" | "XOR" >

UnaryOperator: < "is null" | "is not null" >

IncludeColumns:

Ist dieser Parameter nicht in der Anfrage enthalten, werden immer alle definierten Felder des Objekts zurück geliefert. Da dies relativ viele sein können, kann man die Payload im Response deutlich reduzieren, wenn man hier angibt, welche Felder übermittelt werden sollen. (Aus technischen Gründen werden einige Felder immer übertragen, auch wenn sie nicht explizit hier aufgeführt werden; dies gilt z.B. für die Schlüsselfelder.)

Beispiel:

```
{
  "Take": "20",
  "IncludeColumns": ["Kurztext", "Vk1"]
}
```

ExcludeColumns:

Dieser Parameter ist das Gegenstück zu den IncludeColumns. Statt zu definieren, welche Spalten man haben möchte, kann man hier definieren, welche Spalten weggelassen werden sollen.