Interface concept IS24

eo

WW

Table of contents

[1. Introduction 3](#_Toc445120085)

[1.1 Definitions & Abbreviations 3](#_Toc445120086)

[1.2 Languages 4](#_Toc445120087)

[2. Logic Interface Design 4](#_Toc445120088)

[2.1 Introduction 4](#_Toc445120089)

[2.2 Interface and Integration definition 5](#_Toc445120090)

[2.2.1 Service API - AJAX & Standard HTTP Request 5](#_Toc445120091)

[2.2.2 Caching of service API calls 5](#_Toc445120092)

[2.2.3 Guiding ideas 6](#_Toc445120093)

[2.2.4 Technical solution 6](#_Toc445120094)

[3. Technical Interface Design 7](#_Toc445120095)

[3.1 Summary 7](#_Toc445120096)

[3.2 Overview of business requirements 7](#_Toc445120097)

[3.3 Non-functional requirements 7](#_Toc445120098)

[3.4 Data type definition 7](#_Toc445120099)

[3.4.1 Complex data types 7](#_Toc445120100)

[3.4.2 Scalar types 8](#_Toc445120101)

[3.5 Overall 8](#_Toc445120102)

[3.5.1 Technical Specification 8](#_Toc445120103)

[3.5.2 API Authentication 9](#_Toc445120104)

[3.5.3 API Versioning 9](#_Toc445120105)

[3.5.4 Parameters 9](#_Toc445120106)

[3.5.5 Example request 10](#_Toc445120107)

[3.5.6 Responses 10](#_Toc445120108)

[3.1 General Services 11](#_Toc445120109)

[3.1.1 Application summary / State diagram 11](#_Toc445120110)

[3.1.2 API specification 11](#_Toc445120111)

[3.2 Product advisor (PA) 16](#_Toc445120112)

[3.2.1 Application summary / State diagram 16](#_Toc445120113)

[3.2.2 API specification 18](#_Toc445120114)

[3.3 Online contract application (OVA) 28](#_Toc445120115)

[3.3.1 Application summary / State diagram 28](#_Toc445120116)

[3.3.2 API Specification 29](#_Toc445120117)

# Introduction

## Definitions & Abbreviations

|  |  |
| --- | --- |
| **Term** | **Description** |
| REST | Style of software architecture (Web API design model) for distributed systems |
| CMS | Content Management System, a program that allows publishing, editing and modifying content as well as maintenance from a central interface |
| SIRD | Supplier Infrastructure Requirements Document, contains all relevant KPIs and requirements for the respective infrastructure |
| API | application programming interface, a protocol intended to be used as an interface by software components to communicate with each other |
| OVA | Online contract application (Online-Vertragsabschluss), application provides functionalities to close a utility contract via the internet |
| PA | Product advisor (Produktberater), application provides a product overview / product detail view for the specific post code |
| APS | Ansprechpartnersuche, application searching for a specific KAM or business customers |
| EDG | E.ON Energie Deutschland |
| EBS | E.ON Business Services |
| VG | sales company (Vertriebsgesellschaft), one out of seven sales companies of E.ON in Germany |
| GetAccess | E.ON Authentication Service |
| Mein E.ON | Customer Selfcare portal for private and small/medium business customers |
| SPOC | Sales Partners & Online Campaigns |

## Languages

The document language is defined with English.

# Logic Interface Design

## Introduction

Migrating the website eon.de and its currently implemented applications to the CQ5 platform requires certain interfaces which provide (existing) backend functionalities and information to the CQ5 platform. The applications in scope are:

* Productadvisor (PA)
* Online Contract (OVA)
* Regionalization (for supporting services)

The logical interface design picture provides the general framework for the following chapters where the technical implementation of each interface will be specified in detail.

## Interface and Integration definition

### Service API - AJAX & Standard HTTP Request

The EBS business logic services can be accessed in two ways. Either way is synchronous. All requests are sent from the client directly, there is no communication between CQ5 and the backend.

* AJAX (for the RESTful data-exchange service calls)
* Standard request (for the file-exchange service calls)

A WSO2 API-Manager[[1]](#footnote-1) is placed in front of the EBS services that will proxy all the requests from the client to the backend. This changes only the set of parameters needed for the client to access the backend.

### Caching of service API calls

In order to improve the overall performance of the website and to reduce dependencies to the backend systems a caching could be implemented, if feasible. The caching will store the information gathered by previous calls and re-use the information if the same input parameters for the service are used. For each service call the lifetime of the cache has to be specified which defines the time the data in the cache is valid.

The caching-strategy could be implemented in different ways:

* HTTP Caching-Mechanisms (Request/Response-Header Information, ETags, etc.)
* Response Caching in API Manager
* Backend-Service Caching

In a first approach, caching is not implemented. The results of the performance-test should specify if a caching strategy is needed.

### Guiding ideas

While defining the logical interface design the following guiding ideas have been identified and were used (besides functional requirements) in order to find the best-fitting integration option and interface type:

* Provide solution in a short timeframe but also ensure sustainability
* Implement simple logic into the frontend, in exceptional cases implement logic redundant
* Provide central services if reusability of functionality is required and feasible
* REST will be used for services
* Any communication will be done directly via the client with AJAX or standard requests
* Parameters are submitted via the classic GET-notation (/service?a=b&c=d)

Those guiding ideas can and should be reused for further implementations to evaluate the best-fitting solution.

### Technical solution

The decision about which technology is used to implement the service architecture resulted in a 2-level-solution. The Level 1 will ensure most browsers and devices required by EDG can be supported properly. The Level 2 will ensure all other devices can be supported. The two levels are defined as follows:

|  |  |  |
| --- | --- | --- |
| **Level** | **Solution** | **Leads to supported browsers** |
| 1 | CORS – A mechanism that allows browsers to send AJAX requests to different (sub-) domains. Necessary for CORS to work is that the client always sends a "Origin" header in the request (done by CQ Javascript snippet) exposing itself whereas the server sends a "Access-Control-Allow-Origin" header in the response (done by EBS backend services). | Firefox 3.5+ Chrome 3+ Internet Explorer 10+ Safari 4+ Opera 12+ |
| 2 | Reverse Proxy – located in the Sapient datacenter. A reverse proxy is installed to redirect all AJAX requests to the API-Manager even if the Same-Origin-Policy would cause the browser to deny the request. | All |

Further details regarding to solution options and an insight about the EDG support requirements can be found in the referenced documents.

# Technical Interface Design

## Summary

The chapter describes the services which have to be provided for each application. It defines the technical specification of the "Logical Interface Design" described within chapter 2. Afterwards all specifications will be checked if they match the business requirements.

The services are described as follows:

* Technical Specification
* Parameters
* Example URL
* Response (incl. error scenarios)

## Overview of business requirements

All business requirements are documented in a separate excel file. See the list of referenced documents.

## Non-functional requirements

All non-functional business requirements are processed within a separate excel file. See the list of referenced documents.

## Data type definition

### Complex data types

All used complex data types for the respective service APIs will be defined in a separate excel file. The excel file is available in the workspace. (see the list of referenced documents)

### Scalar types

|  |  |
| --- | --- |
| **Type** | **Description** |
| string | Contains every type of text (calculated or rounded prices are also a string, so that the defined number of decimal places stays in place) |
| boolean | Contains the value "true" or "false" |
| number | Contains a valid integer or floating-point number, decimal separator is always a dot. |
| date | Contains a date with the format "dd.MM.yyyy" |
| datetime | Contains a date with the format "dd.MM.yyyy HH:mm:ss" |
| binary | Contains a binary dataset with like pdf |

Table 1 - Overview of scalar data types

## Overall

This section contains information valid for all services. Generally speaking

* all parameter values must be URL-encoded ("&","Ö")
* parameters can be either set via the path or the request body
* response type is either application/json or application/pdf

### Technical Specification

|  |  |  |
| --- | --- | --- |
| **Caching** | No | See 2.3.2 Caching of service API calls |
| **Authentication API-Manager** | Yes | Authentication is done via access token and a domain token (for the API-Manager) as part of the request headers. |
| **Authentication Backend** | Yes | Authentication is done via clientId as part of the query parameters. |
| **Encryption** | Yes | All traffic is routed via HTTPS |
| **Request-Concept** | AJAX |  |

### API Authentication

For calling an API the Client need to send an access- and a domain-token in the request-header.

* The API-Manager generate (internal) an “SONG”-Application. Every API is assigned to this Application.
* The access-token for this internal Application is generated by the API-Manager for every stage with an unlimited lifetime (e.g. “Bearer a1e18ed661f93bf2e26b932c32608aa9”).
* Because we have three stages for the API-Manager (four for PHP-Endpoints), the GUI-Client needs three Access-Tokens. The Access-Tokens are delivered from the API-Manager Team to the Client-Team.
* The token is needed for all API calls except the services “/pdf” (chapter 3.2.2.6) and “/contractpdf” (chapter 3.3.2.2). The reason for this exception is that pdfs are requested directly by the browser via link, so that you can’t set the header parameter (see [onTrack ticket 290](https://ontrack.intranet.eon-is.com/jira/browse/EDG-P0007252-290)).

### API Versioning

Each API has its own version number. A version number has two digits. It’s possible, but not recommended, that an API exists in different versions at the same time.

The URL scheme looks like this:

<stage>api.eon.de/sales/<api-name>/<version>

Example of a productive request to the /streets service:   
GET https://api.eon.de/sales/streets/1.0?clientId=eonde&zipCode=80335&streetName=a

### Parameters

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Parameter** | **Value(s)** | **Description** | **Parameter type** | **Data type** | **Required** |
| clientId | eonde | A defined identification string of the client | Query | string | X |

### Example request

The example requests do not include the parameters above since they are required for all service calls.

### Responses

|  |  |  |
| --- | --- | --- |
| Status code | Description | Response |
| 200 | OK | See each service separately |
| 400 | Bad request | [  { "parameter": "type",  "code": "ERR\_PARAMETER\_MISSING",  "error-message" : "" },  { "parameter": "filename",  "code": "ERR\_ PARAMETER \_MISSING",  "error-message" : "" } ] |
| 404 | Not found | See each service separately |
| 500 | API-Manager service endpoint is not reachable | <am:fault xmlns:am="http://wso2.org/apimanager">  <am:code>101503</am:code>  <am:type>Status report</am:type>  <am:message>Runtime Error</am:message>  <am:description>Error connecting to the back end</am:description>  </am:fault> |
| 500 | API-Manager accessToken is missing (only if required) | <ams:code>900902</ams:code>  <ams:message>Missing Credentials</ams:message>  <ams:description>Required OAuth credentials not provided</ams:description>  </ams:fault> |
| 500 | Unexpected error within services | [Blank body] |
| 500 | Unexpected, but caught error within services | [  { "parameter": "",  "code": "ERR\_SERVICE",  "error\_message": "Database is not available" } ] |

## General Services

### Application summary / State diagram

The general services are a collection of lightweight services, that most of the time serves as lookup for eon.de related “general” information. For example, for which cities eon.de provides its products or a list of grid operators in Germany.

### API specification

#### /bankname

This service enables the automated completion of the bank name / credit institute in the online contract application.

##### Technical specification

|  |  |  |
| --- | --- | --- |
| **Request type** | GET |  |
| **Response type** | JSON |  |

##### Parameters

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Parameter** | **Value(s)** | **Description** | **Parameter type** | **Data type** | **Required** |
| accountNumber | DE99123456781234567890, 1234567890 | Bank account number provided by the end-customer (IBAN or regular bank account number) | Query | string | X |
| bankCode | BLZ | Bank code, is required if accountNumber does not start with DE | Query | string |  |

##### Example request

GET /bankname?accountNumber=DE63241511160000123456  
GET /bankname?bankCode=79050000

##### Responses

|  |  |  |
| --- | --- | --- |
| Status code | Description | Response |
| 200 | OK | {  "bankname": "Sparkasse Duisburg"  } |
| 404 | Not found | [  { "parameter": "",  "code" : "ERR\_BANKNAME\_NOT\_FOUND",   "error-message": "No bank name for provided parameters found"  } ] |

#### /energyproviders

This service enabled to autocomplete functionality for the "Vorversorger" select box in the online contract application in case of an energy provider change.

##### Technical specification

|  |  |  |
| --- | --- | --- |
| **Request type** | GET |  |
| **Response type** | JSON |  |

##### Parameters

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Parameter** | **Value(s)** | **Description** | **Parameter type** | **Data type** | **Required** |
| name | eon | At least the first three letters of the energy provider | Query | string | X |
| productType | Strom / Erdgas | The product type | Query | string | X |

##### Example request

GET /energyproviders?name=E-WIE&productType=Strom

##### Responses

|  |  |  |
| --- | --- | --- |
| Status code | Description | Response |
| 200 | OK | [  "E.ON Hanse",  "E.ON Avacon",  "Stadtwerke M\u00fcnchen" ] |

#### /streets

This service enables to autocomplete for input field "Straße" for both "Lieferanschrift" and "Alternative Rechnungsanschrift" in the online contract application.

##### Technical specification

|  |  |  |
| --- | --- | --- |
| **Request type** | GET |  |
| **Response type** | JSON |  |

##### Parameters

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Parameter** | **Value(s)** | **Description** | **Parameter type** | **Data type** | **Required** |
| zipCode | 80335 | Complete zip code of the city entered by the end-customer | Query | string | X |
| streetName | a | At least the first character of the street name | Query | string | X |

##### Example request

GET /streets?zipCode=80335&streetName=a

##### Responses

|  |  |  |
| --- | --- | --- |
| Status code | Description | Response |
| 200 | OK | [  "Arnulfstr.",  "Badstr." ] |

#### /zipcodes

Retrieve a list of zip codes for auto-suggestion purposes. The service only returns zip codes for which products are actually available. The city selected by the end-customer needs to be stored in a cookie (CQ5/Frontend).

##### Technical specification

|  |  |  |
| --- | --- | --- |
| **Request type** | GET |  |
| **Response type** | JSON |  |

##### Parameters

No specification for the type ("Sparte") necessary, backend uses electricity ("Strom") as leading type for its functionality.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Parameter** | **Value(s)** | **Description** | **Parameter type** | **Data type** | **Required** |
| ~~zipCode~~ | ~~813~~ | ~~Zip code provided by end-customer (at least 3)~~  **Deprecated** | ~~Query~~ | ~~string~~ | ~~X~~ |
| query | 977 / Karls | Zipcode or parts of it for the lookup or alternatively a city name or parts of it | Query | string | X |
| limit | 10 | The limit of possible autocomplete zipcodes to return | Query | string |  |
| productType | Strom / Erdgas | The type of the product. | Query | String |  |
| customerType | pk / gw | Private customer or small business customer | Query | string |  |

##### Example request

GET /zipcodes?zipCode=06502

##### Responses

|  |  |  |
| --- | --- | --- |
| Status code | Description | Response |
| 200 | OK | {  "zipCodes":  [  {  "zipCode": "06502",  "cities": [  {  "city": "Thale",  "areaType": "G"  },  {  "city": "Timmenrode",  "areaType": "W"  }  ]  },  ],  "more": false  } |

## Product advisor (PA)

### Application summary / State diagram

The product advisor constitutes the sales-related core of the eon.de-site. It serves as the main platform to provide both returning and prospective customers with an overview and details regarding both electricity- and natural gas products for private as well as business customers. It ultimately leads the customer into the online-conclusion process of a contract completed with E.ON. In addition the PA provides the customer with price information in a transparent and comparable manner. The price output is specified through regionalization and the definition of consumption (by entering a postal code and an annual consumption quantity). The product detail sites are also part of the product advisor (PA). The underlying price information will be provided by a fixed, predefined data format.

**SPOC:**

Requests to the product advisor service **/products** will be extended by four additional parameters, i.e. offlineProduct, priceGuarantee, ecology and flexibility. These parameters provide the possibility to reduce the response to the desired products for any campaign/cooperation. For example:

[https://api.eon.de/sales/products/1.0?clientId=eonde&zipCode=97753&city=Karlstadt&](https://api.eon.de/sales/products/1.0?clientId=eonde&zipCode=97753&city=Karlstadt&flexibility=true)**[flexibility=true](https://api.eon.de/sales/products/1.0?clientId=eonde&zipCode=97753&city=Karlstadt&flexibility=true)**

would only return “Flexi” products.

Combinations of these parameters are treated with an AND, e.g. priceGuarantee=true&ecology=true reduces response to products that have both features.

Responses from **all** product advisor **services** will not contain additional fields if the service called is for a sales partner, but rather existing fields for a bonus are recycled, i.e. bonus, bonusUnit, bonusDesc, bonusDescFootnotes, bonusNetEuroRounded and bonusVatEuroRounded. In case the bonus is not a price, e.g. miles the value in bonusVatEuroRounded is of course not rounded or multiplied by the vat factor, but rather will be the same as bonus.

An API call for a sales partner can be identified by the “clientId”. As an example:

[https://api.eon.de/sales/products/1.0?clientId=](https://api.eon.de/sales/products/1.0?clientId=airberlin)**[airberlin](https://api.eon.de/sales/products/1.0?clientId=airberlin)** …

For details please refer to referenced document 2 Complex data types of service API (Field 126 – 130).

**Processes/Interfaces**

* The specific search and display functions will be controlled by the underlying data and prompted by interacting with the CMS
* Change-over to the online contract application
* Postal code-routing for regionalization
* Interface/Transfer private customers/business customers is considered (threshold)
* Delineation towards individual customers (IC); pre-specified annual consumption inputs will be forwarded to the business customer section (electricity: > 100,000kWh; natural gas: > 300,000kWh)
* In case of accessing the OVA, pre-testings in product advisor are conducted. This ensures the successful process flow of the OVA

**State diagram**

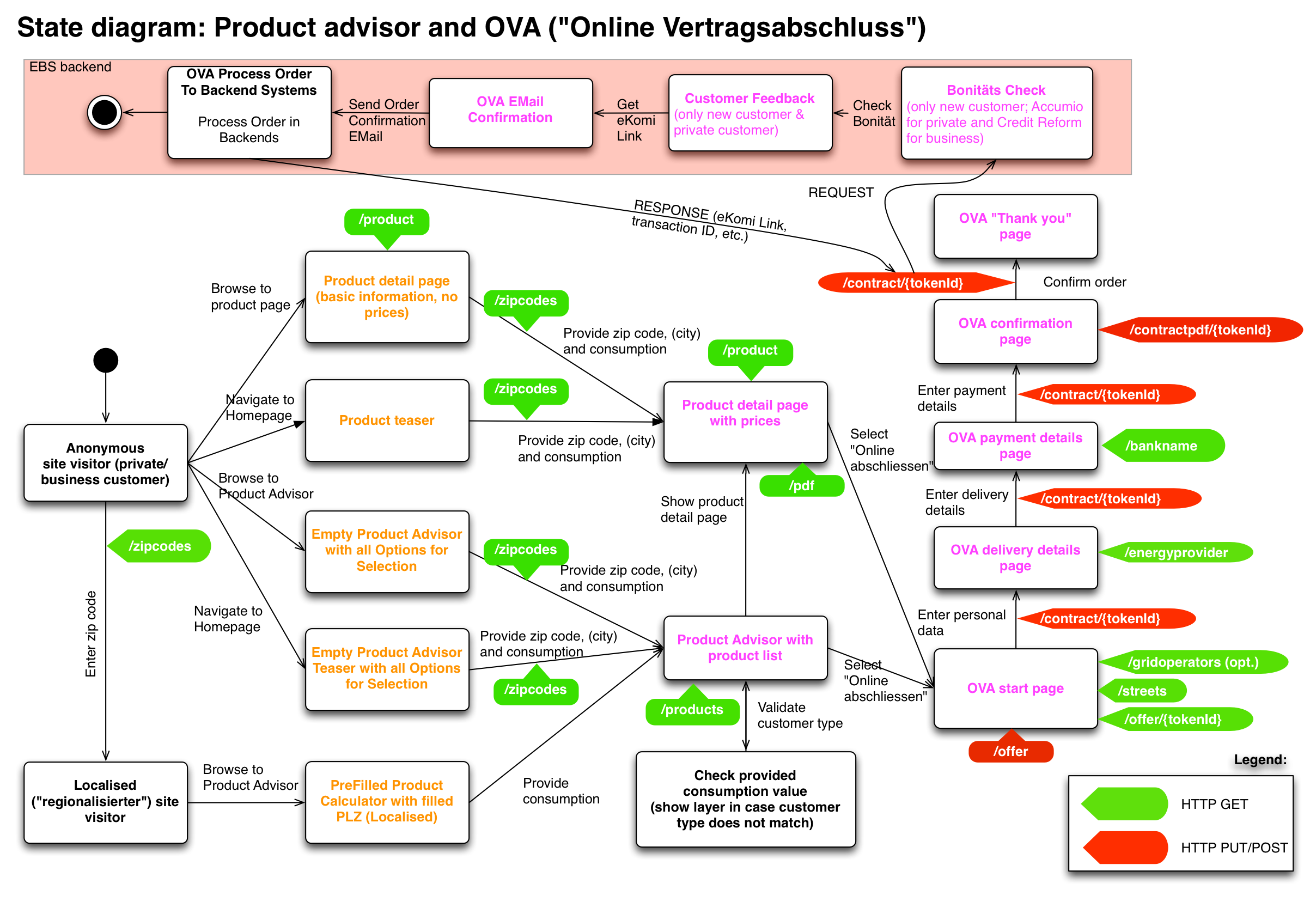


Illustration 1 - State diagram of product advisor

### API specification

#### POST Requests

The POST method is enabled for all PA services in order to be able to pass long parameters, therefore these services are still read only queries. This method was specifically introduced to handle an interim solution for the customer self-care portal concerning product switch & home move, where large amount of user related data had to be transmitted.

##### Technical Specification

|  |  |  |
| --- | --- | --- |
| **Request type** | POST | Content-Type: application/json |
| **Response type** | JSON |  |

##### Parameters

The POST method requires/accepts the same parameters as the GET requests, and in **addition** all services nowaccept following parameters:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Parameter** | **Value(s)** | **Description** | **Parameter type** | **Data type** | **Required** |
| mode | PS | HM | PS = product switch  HM = home move | Query | string |  |
| customerData | eyJ0ZXN0IjoidGVzdCJ9 | Base64 encoded json object. Contains data from the csc that is only used by the EBS API | **Body** | string/json |  |
| user | {  "firstname": "Max",  "lastname":"Mustermann"  } | Personal data from the csc, that is used to prefill certain fields on FE and BE | **Body** | string/json |  |

Table 4 - Input parameters for service POST products

##### Example request

Example for /product:

POST /products?clientid=**csc**&zipCode=80634&productType=Erdgas&consumption=15000&customerType=pk&tariffType=et&percentNt=0.55&signedCustomer=false&**mode=HM**

Content-Type: application/json

|  |
| --- |
| {  "customerData": "eyJ0ZXN0IjoidGVzdCJ9",  "user": {  "firstname": "Max",  "lastname": "Mustermann"  }  } |

##### Responses

Same as the GET request, for details have a look at the “Responses” section for each service.

#### /products

This service allows operations on product lists.

##### Technical Specification

|  |  |  |
| --- | --- | --- |
| **Request type** | GET |  |
| **Response type** | JSON |  |

##### Parameters

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Parameter** | **Value(s)** | **Description** | **Parameter type** | **Data type** | **Required** |
| zipCode | 12345 | Zip code provided by the end-user | Query | string | X |
| city | "München" | In case zip code is not unique, end-user needs to provide the city | Query | string | X |
| productType | Strom / Erdgas | Type of product | Query | string | X |
| consumption | 3000 | The yearly consumption in kWh as integer value | Query | number | X |
| customerType | pk / gw | Private customer or small business customer | Query | string | X |
| tariffType | et / dt | Single tariff or double tariff | Query | string | X |
| percentNt | Default: 0  Example: 0.55 | Percentage of low tariff | Query | number |  |
| signedCustomer | true (signed customers) / false (new customers, default) | Calculate prices for perspective new or signed customers, Configuration of perspective via CMS is possible | Query | boolean |  |
| offlineProduct | true | false | false is default  reduce response to offline products | Query | boolean |  |
| priceGuarantee | true | false | false is default  reduce response to products with price guarantee | Query | boolean |  |
| ecology | true | false | false is default | Query | boolean |  |
| flexibility | true | false | false is default | Query | boolean |  |

Table 5 - Input parameters for service GET products

##### Example request

GET /products?zipCode=80634&productType=Erdgas&consumption=15000&customerType=pk&tariffType=et&percentNt=0.55&signedCustomer=false

##### Responses

|  |  |  |
| --- | --- | --- |
| Status code | Description | Response |
| 200 | OK | See complex data type "product list" |
| 404 | Not found | [  {  "parameter": "",  "code” : "ERR\_PRODUCT\_LIST\_NOT\_FOUND",   "error-message" : "No product list for provided parameters found"  } ] |

#### /product

Retrieve a specific product depending on either the product ID or the product name.

##### Technical specification

|  |  |  |
| --- | --- | --- |
| **Request type** | GET |  |
| **Response type** | JSON |  |

##### Parameters

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Parameter** | **Value(s)** | **Description** | **Parameter type** | **Data type** | **Required** |
| productId | EAV00\_E007135 / <empty> | Productinformation identifier (only for national product displaying) | Query | string | X If not regionalised |
| productName | E.ON DirektStrom | Product identifier | Query | string | X |
| zipCode | 12345 | Zip code provided by the end-user | Query | string |  |
| city | "München" | In case zip code is not unique, end-user needed to provide the city in the frontend | Query | string | X |
| productType | Strom / Erdgas | Type of product | Query | string | X |
| consumption | 3000 | The yearly consumption in kw/h as integer value | Query | number | X |
| customerType | pk / gw | Private customer or small business customer | Query | string | X |
| tariffType | et / dt | Single tariff or double tariff | Query | string | X |
| percentNt | Default: 0  Example: 0.55 | Percentage of low tariff | Query | number |  |
| signedCustomer | true (signed customers) / false (new customers, default) | Calculate prices for perspective new or signed customers, via CMS is possible | Query | boolean |  |

Table 6 - Input parameters for service products/{productId}

##### Example request

GET /product? productId=EAV00\_E007135&productType=Strom&consumption=15000&customerType=pk&tariffType=et&percentNT=0.55&signedCustomer=false  
GET /product? productName=E.ON+%C3%96koStrom&zipCode=80634&productType=Strom&consumption=15000&customerType=pk&tariffType=et&percentNt=0.55&signedCustomer=false

##### Responses

|  |  |  |
| --- | --- | --- |
| Status code | Description | Response |
| 200 | OK | See complex type "ProductAdvisor" |
| 404 | Not found | [  { "parameter": "",  "code" : "ERR\_PRODUCT\_NOT\_FOUND",  "error-message" : "No product with ID/name xyz found"  } ] |

#### /offer

This service is used to start the process of the OVA. It fixates/freezes the current product request within the backend for the complete process of the OVA. The client receives a unique token for the later requests to store contract data. Additionally, see the business requirements document, this service verifies if several legal requirements are fulfilled on the backend side. If this validation fails in the backend, it does not generate a token and prevents the OVA-process to start.

##### Technical specification

|  |  |  |
| --- | --- | --- |
| **Request type** | GET |  |
| **Response type** | JSON |  |

##### Parameters

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Parameter** | **Value(s)** | **Description** | **Parameter type** | **Data type** | **Required** |
| zipCode | 12345 | Zip code provided by the end-user | Query | string | X |
| city | "München" | In case zip code is not unique, end-user needed to provide the city in the frontend | Query | string | X |
| productName | E.ON DirektStrom | Product identifier (all special characters will be delivered unchanged) | Query | string | X |
| productType | Strom / Erdgas | Type of product | Query | string | X |
| consumption | 3000 | The yearly consumption in kWh as integer value | Query | number | X |
| customerType | pk / gw | Private customer or small business customer | Query | string | X |
| tariffType | et / dt | Single tariff or double tariff | Query | string | X |
| gridOperator | Bayernwerk AG | Grid operator selected by the user for product "E.ON ProfiStrom eFix" | Query | string |  |
| percentNt | Default: 0  Example: 0.55 | Percentage of low tariff | Query | number |  |

##### Example request

GET /offer?zipCode=80634&productName=E.ON+%C3%96koStrom&productType=Strom&consumption=15000&customerType=pk&tariffType=et&percentNt=0.55&gridOperator=Bayernwerk+AG

##### Responses

|  |  |  |
| --- | --- | --- |
| Status code | Description | Response |
| 200 | OK | {  "tokenId": "klghe54k3e58oudhdfklghe89z" } |
| 404 | Not found | [  { "parameter": "",  "code" : "ERR\_PRODUCT\_NOT\_FOUND",  "error-message" : "No product with ID xyz found"  } ] |
| 500 | Internal Server Error | [  { "parameter": "",  "code": "ERR\_AGB\_DISTINFO\_INVALID",  "error-message": "Either the AGB file or DistInfo are missing"  } ] |

#### /offer/{tokenId}

This service with an additional tokenId parameter provides the possibility for the client, to fetch the product data which was earlier fixated / frozen in the backend.

##### Technical specification

|  |  |  |
| --- | --- | --- |
| **Request type** | GET |  |
| **Response type** | JSON |  |

##### Parameters

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Parameter** | **Value(s)** | **Description** | **Parameter type** | **Data type** | **Required** |
| tokenId | klghe54k3e58oudhdfklghe89z | Token generated by the EBS backend to fixate the product data for the OVA | Query | string | X |

##### Example request

GET /offer?tokenId=klghe54k3e58oudhdfklghe89z

##### Responses

|  |  |  |
| --- | --- | --- |
| Status code | Description | Response |
| 200 | OK | See complex type "ProductAdvisor" |
| 404 | Not found | [  { "parameter": "",  "code" : "ERR\_TOKEN\_NOT\_FOUND",  "error-message" : "No token with ID xyz found. "   } ] |

#### /pdf

This service is used to retrieve PDF files. The CQ5 frontend displays links it builds with the data received by other services. Because the AJAX-standard cannot be used here, the service can be called directly without authentication header. This is not a security issue, because it is a public document (“AGB”s).

In case of errors (“file not found”) it returns a Redirect-Header to force an error page within the CQ5 while breaking the RESTful and independent logic with that.

##### Technical specification

|  |  |  |
| --- | --- | --- |
| **Request type** | GET |  |
| **Response type** | Binary data | Mime-Type: application/pdf |
| **Request-Concept** | Standard request |  |

##### Parameters

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Parameter** | **Value(s)** | **Description** | **Parameter type** | **Data type** | **Required** |
| type | agb / gvv / auftrag / tarifbeiblatt / preiszusammensetzung | agb / gvv / auftrag / tarifbeiblatt / preiszusammensetzung | Query | string | X |
| filename | EHA00\_E009132.pdf | Frontend must generate target filename:  agb filename pattern: AGB\_[Type]\_[salesArea]\_[optional:Gewerbe].pdf,  auftrag filename pattern:  [productId].pdf, for example: EHA00\_E009132.pdf, Exception: Heizstrom AGB for business customers must not use suffix Gewerbe  gvv filename pattern: electricity - https://www.eon.de/resources/base/docs/pdf/Produktinformationen/gvv/StromGVV.pdf  gas - https://www.eon.de/resources/base/docs/pdf/Produktinformationen/gvv/GasGVV.pdf | Query | string | X |
| errorUrl | <https://www.eon.de/file_not_found.html> | URL of the error page the service sends a Redirect-Header to in case of “File not found” | Query | string | X |

##### Example request

GET /pdf?type=auftrag&errorUrl= https%3A%2F%2Fwww.eon.de%2Ffile\_not\_found.html &filename=EHA00\_E009132.pdf  
GET /pdf?type=agb&errorUrl=https%3A%2F%2Fwww.eon.de%2Ffile\_not\_found.html &filename=AGB\_Erdgas\_EHAV\_Gewerbe.pdf  
GET /pdf?type=gvv&errorUrl=https%3A%2F%2Fwww.eon.de%2Ffile\_not\_found.html &filename=StromGVV.pdf

##### Responses

|  |  |  |
| --- | --- | --- |
| Status code | Description | Response |
| 200 | OK | [Binary data] |
| 307 | Temporary Redirect | Additional Response-Header: Location: {errorUrl}” |

## Online contract application (OVA)

### Application summary / State diagram

As additional part of the sales-related core of the PA, the OVA enables the customer/prospect to conclude a contract without media disruption.

**Content depiction**

The OVA is a generic application enabling a flexible use for web-based applications. The process flow, beginning with Step1 and concluding with Step4, is comparable within all product ranges. Differences exist with regard to the specification of private- and business customers and due to the product divisions.

**Processes/Interfaces**

* Connectivity to backend-systems (ERDS/CRM) for further processing of contract data is carried out through the ERDS-contract conclusion entry-interface (CR843). The entry-interface will be used for both online contract conclusions via eon.de as well as for other sales channels
* Credit assessment via interface is conducted by one external service providers. Soon, the credit assessment will be carried out within the backend-system (ERDS/CRM, CR997); therefore, the implementation within eon.de can be regarded as an interim solution
* Customer reviews are conducted through an external service provider

### API Specification

#### /contractpdf/{filename}

This service is used to retrieve the printable version of the contract pdf. The filename format consists of the normalized product name (no spaces, umlauts or dots) and a static text.

Example: EONDirektStrom\_zusammenfassung\_auftragsdaten\_druckansicht.pdf

Because the AJAX-standard cannot be used here, the service can be called directly without authentication header. This is not a security issue, because it is protected by a user-specific tokenId.

##### Technical specification

|  |  |  |
| --- | --- | --- |
| **Request type** | GET |  |
| **Response type** | Binary data | Mime-Type: application/pdf |
| **Request-Concept** | Standard request |  |

##### Parameters

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Parameter** | **Value(s)** | **Description** | **Parameter type** | **Data type** | **Required** |
| tokenId | 12345 | Token generated by the EBS backend to fixate the product data for the OVA | Path | string | X |
| confirmation\_advertising\_mail | 0 / 1 | Confirmation of the end-customer to agree (1) or disagree (0) to advertising by mail | Query | number | X |
| confirmation\_advertising\_phone | 0 / 1 | Confirmation of the end-customer to agree (1) or disagree (0) to advertising by phone | Query | number | X |

##### Example request

GET /contractpdf?filename=EONDirektStrom\_zusammenfassung\_auftragsdaten\_druckansicht.pdf& tokenId=klghe54k3e58oudhdfklghe89z&confirmation\_advertising\_phone=1&confirmation\_advertising\_mail=0

##### Responses

|  |  |  |
| --- | --- | --- |
| Status code | Description | Response |
| 200 | OK | [Binary data] |

#### /contract/{tokenId}

This service is used to submit contract data (via the OVA form) for the given token ID. The frontend will invoke this service operation for each OVA step together with the generated token ID and submit all personal data provided by the end-customer. To support flexible usage of the service and keep the service stateless, the fixated/frozen product data plus the already entered and validated user data are returned on each successful response.

**SPOC:**

The API call doesn’t change since all step info’s are sent via the parameter “data”. Though adding additional fields to the request need to be arranged with the backend first, so that respective validators can be adapted or added to guarantee correct behavior.

EDG should also be contacted in case new error codes need to be defined.

An API call for a sales partner can be identified by the “clientId”. As an example:

[https://api.eon.de/sales/contract/1.0?clientId=](https://api.eon.de/sales/contract/1.0?clientId=airberlin)**[airberlin](https://api.eon.de/sales/contract/1.0?clientId=airberlin)** …

##### Technical specification

|  |  |  |
| --- | --- | --- |
| **Request type** | POST |  |
| **Response type** | JSON | string/json |

##### Parameters

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Parameter** | **Value(s)** | **Description** | **Parameter type** | **Data type** | **Required** |
| tokenId | klghe54k3e58oudhdfklghe89z | Token generated by the EBS backend to fixate the product data for the OVA | Query | string | X |
| step | 1 | 2 | 3 | 4 | Number of the step the respective data is submitted for – necessary for primarily the required” validations | Body | string | X |
| data | {"salutation": "Herr",  "firstname": "Max",  "lastname": "Mustermann",  "birthday": "10.05.1971"…}  See detailed description (eonde\_complex\_datatypes.xls) | Form data submitted for the respective form step (1-4). The “data” contains all step parameters in a Json structure. | Body | string | X |
| mode | PS | HM | PS = product switch  HM = home move | Query | string |  |

##### Example request

POST /contract?tokenId=klghe54k3e58oudhdfklghe89z&clientId=eonde HTTP/1.1  
 "step" = 1,  
 “data” = “{  
 "salutation": "Herr",  
 "firstname": "Max",  
 "lastname": "Mustermann",  
 "birthday": "10.05.1971",  
 [more form data]

}”

##### Responses

|  |  |  |
| --- | --- | --- |
| Status code | Description | Response |
| 200 | OK – step 1 – 3 | {  "products": [  {See complex type ProductAdvisor: }  ],  "footnotes": {},  "ova": [  [  {  "name": "Mustermann"  },  {  "firstname": "Max"  }  ],  [  {  "contractNumber": "123456"  }  ]  ] } |
| 200 | OK – step 4 | {  "contractId": "123456",  "ekomiLink": "<https://www.ekomi.de/...>",  "product": [  {See complex type ProductAdvisor: }  ],  "ova": [  [  {  "name": "Mustermann"  },  {  "firstname": "Max"  }  ],  [  {  "contractNumber": "123456"  }  ]  ] } |
| 400 | Bad request | [  { "parameter": "accountNumber",  "code": "ERR\_IBAN\_FALSE",   "error-message": "Provided IBAN is invalid" },  { "parameter": "birthday",  "code": "ERR\_BIRTHDAY\_INVALID",   "error-message": "Customer must at least be 18 years old" } ] |

1. http://wso2.com/products/api-manager/ [↑](#footnote-ref-1)