Guru cue Ltd.
Tržaška c. 126, SI-1000 Ljubljana
info@gurucue.com
http://gurucue.com

GuruCue Search & Recommendation Engine

RESTful Web Interface

Ljubljana, 2017, v23

gurucue

Table of Contents

1 Introduction

RESTful web services are based on the REST architecture using the HTTP protocol. In essence this means that clients use the HTTP methods GET, POST, PUT and DELETE for sending requests to a server, where the request body (with methods POST and PUT) is presented in the XML or JSON format. Which method to use is in general subject to what is intended, or what function is being invoked:

- GET is generally used to fetch data about an entity,
- PUT is generally used to create a new or replace an existing entity,
- POST is generally used to modify an existing entity,
- DELETE is generally used to remove an entity.

It is not required for a web service to implement all of the methods; these are only recommendations.

A response represents a success if the HTTP response code is 200 and the resultCode of the body is between 0 and 999. In case of an error there are two possible types of response:

- HTTP code different from 200, typically 500, means a failure in the service operation,
- HTTP code 200 with resultCode in body of the response 1000 or greater means an error in the request format or request data.

The recommendations system implements the following services:

- Product
- Consumer
- Event
- Recommendation
- Search
- Health

1.1 Consumer Events

Consumers generate events by interacting with devices and using services, e.g. set-top boxes, web services. These events can in general be anything, from simple menu navigation to rating a movie. Some of these events are interesting for the GuruCue system to acquire a better representation of an individual's taste, so it can consequently create better personalised recommendations. The interesting events to the GuruCue system are the events that contain data about a consumer's activity in respect to certain content, e.g. displaying a movie description from a list of movies, rating a movie, etc.

To describe when a certain consumer did something in connection with a certain movie, the GuruCue system needs to receive a consumer event, which must contain at least the following pieces of data:

- the timestamp when the event occured,
- the event type which defines the type of activity that the consumer executed (or: what did the consumer do),
- the identifier of the consumer that triggered the event,
- the identifier of the content related to the event.

Usually we can say a bit more about an event when it occurs, than just when and what a certain consumer did with a certain content. For example: when a certain consumer starts to watch a movie we usually know what type of device the consumer is using and what menu/category/page did consumer use to reach the movie and initiate playback, or when a consumer rates a movie the actual rating is also a component of the event. These additional pieces of information are termed event descriptors as they contain additional data to further describe an event.

1.1.1 Consumer Event Descriptors

An event can contain any number of event descriptors, theoretically there is no limit to which and how many descriptors an event can contain. For practical reasons the GuruCue system imposes certain limits: for each event type it is configured to take into account only certain descriptors. Therefore it is unnecessary to include descriptors other than what GuruCue actually uses. In other words: for each event type the GuruCue system stores only descriptors that are deemed useful, which is also a guide to define possible descriptors for each event type. The possibility to use a descriptor to enhance recommendations is not the only usefulness of a descriptor, a descriptor can be useful for analyses, too. Exactly analyses are the tools that usually determine if and how can a certain descriptor be used to enhance the quality of recommendations. From this point of view we are always open for suggestions from our partners about what descriptors they can provide.

2 Products

In a broader context products are divided into service-based and content-based products. Service-based products are services provided by an operator (e.g. subscription to a TV package), while content-based products are the content that is offered as part of a service (e.g. a subscription) and/or individually (e.g. a movie purchase).

The recommendation system requires a more specific division, so there is a number of product types according to their semantics, which are listed below.

Product type name	**Description**
package	Subscription package
video	Video content: movies, documentaries, ...
tv-programme	An EPG element, a show aired at a specific time on a specific TV-channel
tv-channel	A TV-channel, having live content
tvod	Transaction VoD catalogue
svod	Subscription VoD catalogue
interactive	Interactive content, a set-top box app

Each product has its own unique ID relative to its product type (numbering space), and its own list of attributes, which at a minimum contains its title or description. Attributes vary according to:

- arity: some attributes allow for only one value (e.g.: title), some allow for any number of values (e.g.: actor),
- translatability: some attributes require that a language also be specified with the value and allow for additional translations to other languages (e.g.: title), with other attributes specifying a language makes no sense therefore they forbid it (e.g.: movie runtime).

Attribute	**Multi-valued**	**Product types it can appear in**	**Translatable**
title	no	video, tv-programme, package, tv-channel, tvod, svod, interactive	yes
tv-channel	yes	tv-programme	no
production-year	no	video, tv-programme	no
country	yes	video, tv-programme	no
video-category	no	tv-programme	no
genre	yes	video, tv-programme	no
season-number	no	video, tv-programme	no
episode-number	no	video, tv-programme	no
director	yes	video, tv-programme	yes
actor	yes	video, tv-programme	yes
run-time	no	video	no
is-adult	no	video	no
spoken-language	yes	video, tv-programme, tv-	no

		channel	
begin-time	no	tv-programme	no
end-time	no	tv-programme	no
video-format	no	video, tv-programme	no
package-type	no	package	no
price	no	video	no
subtitle-language	yes	video, tv-channel	no
video-id	no	tv-programme	no
catchup-hours	no	tv-channel	no
tv-channel-id	yes	package	no
svod-id	yes	package	no
interactive-id	yes	package	no
catalogue-id	yes	video, tvod, svod	no
imdb-link	no	video, tv-programme	no
imdb-rating	no	video, tv-programme	no
parental-rating	no	video, tv-programme	no
description	no	video, tv-programme	yes
image-url	no	video, tv-programme	no
vod-category	yes	video	no
validity	yes	video	no

Product management must be synchronous with the changes in the CMS.

2.1 Packages (package)

A package describes a subscription package, which can contain tv-channels, SVOD catalogues, and interactive apps.

Attribute	Description
title	Package name
package-type	Package type (tv-channel package, supplement, ...)
tv-channel-id	Product ID of a tv-channel type product
svod-id	Product ID of a svod type product
interactive-id	Product ID of an interactive type product

TV-channels contained by the package are given with the tv-channel-id attribute: one attribute instance per TV-channel. Similarly it applies to SVOD (attribute svod-id) and interactive (attribute interactive-id) products. Packages may contain mixed subscriptions, e.g.: a few tv-channels and a few SVOD catalogues.

2.2 Video Content (video)

Video content, tv-programme content, and events form the basis of the recommendation system. Video content is assigned to TVOD and SVOD catalogues. There are many attributes that can be set to video content.

Attribute	Description

title	Movie title
production-year	Production year
country	Country of the movie
genre	Movie genre
director	Movie director
actor	Movie actor
run-time	Movie runtime, in minutes
is-adult	Whether it is adult-only: true, false
spoken-language	Spoken language in the movie
video-format	Picture format: sd, 720p, 1080p
price	Movie price if it is chargeable
subtitle-language	Subtitle language
catalgoue-id	Catalogue code
season-number	Season number, for episodes of a tv-series
episode-number	Episode number, for episodes od a tv-series
imdb-link	URL to or ID of the content at IMDB
imdb-rating	Rating of the content at IMDB, as a number between 1 and 10
parental-rating	Minimum acceptable viewer age, in years of age
vod-category	VOD Category name
validity	A couple of timestamps separated by a space: content validity start and end

2.3 TV-programme Shows (tv-programme)

An electronic programme guide (EPG) is composed of tv-shows, where this means any types of broadcast, such as: movies, tv-series, sport events, news shows, etc. Generally tv-programme items can be considered as an extension of video items, with added attributes related to TV-broadcasting.

Attribute	**Description**
title	TV-show title
tv-channel	TV-channel ID where the show is aired
production-year	Production year
country	Country of origin
video-category	Video category name
genre	Genre
season-number	Season number, for episodic content
episode-number	Episode number, for episodic content
director	Director
actor	Actor
is-adult	Whether it is adult-only: true, false
spoken-language	Spoken language in the TV-show
begin-time	Airing start, either as the unix timestamp (number of seconds since 1970-01-01 00:00:00 UTC), or in the format YYYY-MM-DD hh:mm:ss[+/-hh:mm]

end-time	Airing end, either as the unix timestamp (number of seconds since 1970-01-01 00:00:00 UTC), or in the format YYYY-MM-DD hh:mm:ss[+/-hh:mm]
video-format	Picture format: sd, 720p, 1080p
imdb-link	URL to or ID of the content at IMDB
imdb-rating	Rating of the content at IMDB, as a number between 1 and 10
parental-rating	Minimum acceptable viewer age, in years of age

2.4 TV-channels (tv-channel)

A tv-channel product describes a TV-channel. If a subtitle-language or a spoken-language is defined then all tv-programmes that belong to the TV-channel inherit this attribute, unless overridden at the tv-programme.

Attribute	Description
title	TV-channel name
subtitle-language	Subtitle language
spoken-language	Spoken language
catchup-hours	Number of hours of catch-up available for this TV-channel
is-adult	Whether this is an adult-only TV-channel: true, false

2.5 TVOD (tvod)

A tvod product type describes a TVOD (transactional video-on-demand) catalogue. The attribute catalogue-id must be set, so video products belonging to the catalogue can be identified.

Attribute	Description
title	TVOD name
catalogue-id	Catalogue ID/number

2.6 SVOD (svod)

A svod product type describes a SVOD (subscription video-on-demand) catalogue. The attribute catalogue-id must be set, so video products belonging to the catalogue can be identified.

Attribute	Description
title	SVOD name
catalogue-id	Catalogue ID/number

2.7 Interactive Content (interactive)

An interactive product type denotes an interactive content, a set-top box app. It doesn't require any specific attributes.

Attribute	Description
title	Name of the interactive content

2.8 Product Management

REST API with HTTP methods PUT, POST and DELETE is used to manage products, at the relative URL /rest/product. The method GET is used to retrieve the current state of a product.

2.8.1 Adding or Replacing a Product

To add or replace a product the HTTP method PUT is used. The request must contain the product type, the product ID, and a list of attributes. All translatable attributes should contain a language code to identify the language in which the value is stated, and should. A language code can be either per ISO-639-1 (a 2-letter code) or ISO-639-2/t (a 3-letter code) standard. If there are translations in other languages available, then a translatable attribute should include the 'translations' element and include them therein. If the language code of a translatable value is missing, then the default code 'unk' (unknown) is used. Translations can not be missing a language code or a translated value.

Element	Description
type	Product type
id	Product ID
attributes	List of attributes

Below is an example in XML.

```
<request>
  <type>video</type>
  <id>45678</id>
  <attributes>
    <attribute>
      <identifier>title</identifier>
      <value>Star Trek Into Darkness</value>
      <language>eng</language>
      <translations>
        <translation>
          <language>slv</language>
          <value>Zvezdne steze: V temo</value>
        </translation>
      </translations>
    </attribute>
    <attribute>
      <identifier>production-year</identifier>
      <value>2013</value>
    </attribute>
    <attribute>
      <identifier>genre</identifier>
      <value>sci-fi</value>
    </attribute>
    <attribute>
      <identifier>run-time</identifier>
      <value>133</value>
    </attribute>
    <attribute>
      <identifier>director</identifier>
      <value>J. J. Abrams</value>
      <language>eng</language>
    </attribute>
```

```
    <attribute>
      <identifier>actor</identifier>
      <value>John Cho</value>
      <language>eng</language>
    </attribute>
    <attribute>
      <identifier>actor</identifier>
      <value>Benedict Cumberbatch</value>
      <language>eng</language>
    </attribute>
    <attribute>
      <identifier>actor</identifier>
      <value>Alice Eve</value>
      <language>eng</language>
    </attribute>
    <attribute>
      <identifier>actor</identifier>
      <value>Bruce Greenwood</value>
      <language>eng</language>
    </attribute>
    <attribute>
      <identifier>actor</identifier>
      <value>Simon Pegg</value>
      <language>eng</language>
    </attribute>
    <attribute>
      <identifier>actor</identifier>
      <value>Chris Pine</value>
      <language>eng</language>
    </attribute>
    <attribute>
      <identifier>actor</identifier>
      <value>Zachary Quinto</value>
      <language>eng</language>
    </attribute>
    <attribute>
      <identifier>actor</identifier>
      <value>Zoe Saldana</value>
      <language>eng</language>
    </attribute>
    <attribute>
      <identifier>actor</identifier>
      <value>Karl Urban</value>
      <language>eng</language>
    </attribute>
    <attribute>
      <identifier>actor</identifier>
      <value>Peter Weller</value>
      <language>eng</language>
    </attribute>
    <attribute>
      <identifier>actor</identifier>
      <value>Anton Yelchin</value>
      <language>eng</language>
    </attribute>
    <attribute>
      <identifier>catalogue-id</identifier>
      <value>8</value>
    </attribute>
  </attributes>
```

</request>

Below is an example in JSON.


```
{
  "type": "video",
  "id": "45678",
  "attributes": [
    {
      "identifier": "title",
      "value": "Star Trek Into Darkness",
      "language": "eng",
      "translations": [
        {
          "language": "slv",
          "value": " Zvezdne steze: V temo"
        }
      ]
    },
    {
      "identifier": "production-year",
      "value": 2013
    },
    {
      "identifier": "genre",
      "value": "sci-fi"
    },
    {
      "identifier": "run-time",
      "value": 133
    },
    {
      "identifier": "director",
      "value": "J. J. Abrams",
      "language": "eng"
    },
    {
      "identifier": "actor",
      "value": "John Cho",
      "language": "eng"
    },
    {
      "identifier": "actor",
      "value": "Benedict Cumberbatch",
      "language": "eng"
    },
    {
      "identifier": "actor",
      "value": "Alice Eve",
      "language": "eng"
    },
    {
      "identifier": "actor",
      "value": "Bruce Greenwood",
      "language": "eng"
    },
    {
      "identifier": "actor",
      "value": "Simon Pegg",
      "language": "eng"
    },
```

```
    {
      "identifier": "actor",
      "value": "Chris Pine",
      "language": "eng"
    },
    {
      "identifier": "actor",
      "value": "Zachary Quinto",
      "language": "eng"
    },
    {
      "identifier": "actor",
      "value": "Zoe Saldana",
      "language": "eng"
    },
    {
      "identifier": "actor",
      "value": "Karl Urban",
      "language": "eng"
    },
    {
      "identifier": "actor",
      "value": "Peter Weller",
      "language": "eng"
    },
    {
      "identifier": "actor",
      "value": "Anton Yelchin",
      "language": "eng"
    },
    {
      "identifier": "catalogue-id",
      "value": "8"
    }
  ]
}
```


2.8.2 Modifying a Product's Data

Using the HTTP method POST attributes of an existing product can be modified. The request is structurally similar to the request for adding a product, with the exception that instead of a single list of attribute there can be two lists of attributes: 'attributes-set' to list the attributes to replace or add, and 'attributes-clear' to list the attributes to remove. It is not considered an error to remove an unexisting product. Note that in the case that an attribute is multi-valued, an existing value cannot be replaced, because a new value will simply be added to any existing values. To replace a value in such a case, the existing value must be listed among the values to remove, and the new value listed among the values to add. With single-valued attributes, a new value always replaces any existing value.

Element	Description
type	Product type
id	Product ID
attributes-set	Attributes to add or replace
attributes-clear	Attributes to remove

Below is an example in XML.

```
<request>
  <type>tv-programme</type>
  <id>34556278</id>
  <attributes-set>
    <attribute>
      <identifier>begin-time</identifier>
      <value>1373634900</value>
    </attribute>
    <attribute>
      <identifier>end-time</identifier>
      <value>1373636700</value>
    </attribute>
  </attributes-set>
</request>
```

Below is an example in JSON:

```
{
  "type": "tv-programme",
  "id": "34556278",
  "attributes-set": [
    {
      "identifier": "begin-time",
      "value": 1373634900
    },
    {
      "identifier": "end-time",
      "value": 1373636700
    }
  ]
}
```

2.8.3 Deleting a Product

To delete a product we use the HTTP method DELETE, with the request path formed from the product type followed by the product ID. For example:

/rest/product/video/21744

2.8.4 Retrieving Product Data

To retrieve current product data use the HTTP method GET, with the request path formed from the product type followed by the product ID. The returned response is structurally the same as the request structure for adding a product. A request path example:

/rest/product/video/21744

3 Consumers

Consumer management must be synchronous with the changes in CMS.

The properties such as the user's personal data are not required for the recommendation system. They do however represent potential dimensions of user segmentation for marketing rules, experimenting, and analytics.

A consumer can have relationships to certain products, e.g. subscriptions to program packages, and followed content ("My favourite shows"). These relationships must be described to the recommendation system for it to work correctly.

Relationship element	**Description, possible values**
product-type	package, tv-programme
product-id	product ID
relation-type	subscription, follow
relation-start	timestamp, either as the unix timestamp (number of seconds since 1970-01-01 00:00:00 UTC), or in the format YYYY-MM-DD hh:mm:ss[+/-hh:mm]
relation-end	timestamp, either as the unix timestamp (number of seconds since 1970-01-01 00:00:00 UTC), or in the format YYYY-MM-DD hh:mm:ss[+/-hh:mm]

Certain relationship types are sensible only with certain product types.

Type of relationship	**Product types it can contain**
subscription	package
follow	tv-programme

There can be multiple instances of the same relationship (defined with "product-type" and "product-id") that differ only in start and end of the relationship (defined with "relation-start" and "relation-end"), where the relationship intervals do not intersect and only one instance of the relationship (the currently active relationship) may lack the element "relation-end".

To create recommendations the system uses consumer history represented with events. To reset his recommendations a consumer is able to delete his history.

Consumers are managed through the REST API with the path /rest/consumer, using HTTP methods POST, PUT and DELETE. To retrieve a consumer's current data the recommendation system provides the HTTP method GET.

In cases where the recommendation system receives a recommendation request or an event referencing an unknown consumer, the recommendation system automatically adds the consumer. This way the recommendation system is able to collect events about the consumer and serve recommendations, though the recommendations may be abridged or incorrect, and segmentation is not possible, until the consumer's data is added.

3.1 Adding a Consumer

A consumer is added or replaced with the HTTP method PUT. The request body must contain the consumer ID and current (with "relation-end" missing or in the future) and possibly any past relationships ("relation-end" in the past) to products.

Property	Description
user-id	Consumer ID
product-relations	A list of relationships
delete-history	Whether to remove any existing history, a boolean ("true" or "false").

Below is an example in XML where we add a new consumer having one past subscription and one active subscription to a package, deleting any history about the consumer.

```
<request>
  <user-id>user-test</user-id>
  <product-relations>
    <relation>
      <product-type>package</product-type>
      <product-id>package-test</product-id>
      <relation-type>subscription</relation-type>
      <relation-start>1301094000</relation-start>
      <relation-end>1320706800</relation-end>
    </relation>
    <relation>
      <product-type>package</product-type>
      <product-id>package-test2</product-id>
      <relation-type>subscription</relation-type>
      <relation-start>1320706800</relation-start>
    </relation>
  </product-relations>
  <delete-history>true</delete-history>
</request>
```

Below is the same request in JSON.

```
{
  "user-id": "user-test",
  "product-relations": [
    {
      "product-type": "package",
      "product-id": "package-test",
      "relation-type": "subscription",
      "relation-start": 1301094000,
      "relation-end": 1320706800
    },
    {
      "product-type": "package",
      "product-id": "package-test2",
      "relation-type": "subscription",
      "relation-start": 1320706800
    }
  ],
  "delete-history": true
}
```

3.2 Changing Data About a Consumer

Using the HTTP method POST we can change data about an existing consumer. The request body is structurally same to the request body of a request to add a consumer.

3.2.1 Following a Content (Consumer's Favourite Shows)

Below is a request body in XML to set following a show for a consumer; the timestamp of when the consumer chose to follow the show is assigned to "relation-start".

```
<request>
  <user-id>user-test</user-id>
  <product-relations>
    <relation>
      <product-type>tv-programme</product-type>
      <product-id>tv-programme-test</product-id>
      <relation-type>follow</relation-type>
      <relation-start>1320706800</relation-start>
    </relation>
  </product-relations>
</request>
```

Below is the same example in JSON.

```
{
  "user-id": "user-test",
  "product-relations": [
    {
      "product-type": "tv-programme",
      "product-id": "1227483450",
      "relation-type": "follow",
      "relation-start": 1320706800
    }
  ]
}
```

When the consumer stops following the show a similar request is sent, with included "relation-end" that contains the timestamp of when the consumer chose to stop following the show. The "relation-start" can be omitted, in this case the currently active relationship (that lacks "relation-end") will be terminated.

3.2.2 Removing Existing History of a Consumer

Consumer's recommendations can be reset by removing his history. This is easiest done with the POST method, providing only the consumer ID and the "delete-history" flag.

Below is an XML example.

```
<request>
  <user-id>user-test</user-id>
  <delete-history>true</delete-history>
</request>
```

Below is the same request in JSON.

```
{
  "user-id": "user-test",
  "delete-history": true
}
```

3.3 Deleting a Consumer

A consumer can be deleted using the HTTP method DELETE, with the consumer ID at the end of the service's path. Below is an example of a path.

/rest/consumer/user-test

3.4 Retrieving Data About a Consumer

The HTTP method GET is used to retrieve data about a consumer by providing the consumer ID at the end of the service's path. The response body is structurally the same as the request body for adding a consumer. Below is an example of a path.

/rest/consumer/user-test

4 Consumer Profiles

The recommendation system supports consumer profiles. They are not essential to the system itself, though if they are provided the system can better handle personalization of recommendations. When events and recommendation requests specify the profile of the consumer on behalf of which the request is sent, then those events are assigned to his/her profile, and recommendations are assembled based on specifics of his/her profile. Otherwise an internal basic profile is used, where it is not known who specifically is targeted in the consumer's household.

Consumer profiles are managed through the REST API with the path /rest/userprofile, using HTTP methods PUT and DELETE. To see if and what profiles are set for a certain consumer, the consumer data must be retrieved: if there is at least one consumer profile set, then the "user-profile-ids" field is set with an array of consumer profile IDs.

4.1 Adding a consumer profile

A consumer profile is set with the HTTP method PUT. The request body must contain the consumer ID for which the consumer profile is set, and the profile ID.

Property	Description
user-id	Consumer ID for which the profile is added.
user-profile-id	Profile ID that is added.

Below is an example in XML where we set a new profile to a consumer.

```
<request>
<user-id>user-test</user-id>
<user-profile-id>profile-test</user-profile-id>
</request>
```

Below is the same example in JSON.

```
{
   "user-id": "user-test",
   "user-profile-id": "profile-test"
}
```

4.2 Deleting a consumer profile

A consumer profile can be deleted using the HTTP method DELETE, with both the consumer ID and profile ID at the end of the service's path. Below is an example of a path.

```
/rest/userprofile/user-test/profile-test
```

5 Events

To serve good recommendations the recommendation system requires information about consumers' activities (clicks, searches, views, purchases), expressed views and opinions about content (ratings, "likes"), and about consumers' social and conceptual links. The system receives this information with events.

Events need to be delivered in real-time, right after they occured. This way events immediately influence creation of recommendations for consumers.

The partner should provide for lossles delivery of events: in case that an event could not be delivered (e.g. a fault on the transport path) it should be resent.

Below is a list of elements that each event should contain.

Element	Description
timestamp	Timestamp signifying the moment the event occured (momentary or short-duration events), or the moment an activity started, either as the unix timestamp (number of seconds since 1970-01-01 00:00:00 UTC), or in the format YYYY-MM-DD hh:mm:ss[+/-hh:mm]
user-id	Consumer ID, as a general rule it is hashed to conceal identity for privacy reasons
product-type	Product type: video, tv-channel, tv-programme, package
product-id	Product ID
type	Type of event: consumption, purchase, zap, rating, interaction
data	A list of event descriptors

As a general rule every event carries with itself additional data which describes it in more detail. This is needed to improve the recommendation system operation and/or to run certain analytics and for evaluation of the system operation. The additional data is represented as a list of event descriptors. Every event type is assigned a set of possible event descriptors.

Event descriptor	Description
device-id	The device ID (MAC address or similar) where the event originated from, usually hashed for privacy reasons
device-type	The type of device where the event originated from
catalogue-id	VOD catalogue ID in connection with the event (purchase, rating, playout)
was-purchased	Whether the content was purchased beforehand: "true" or "false"
watch-offset	Position on the content timeline as an offset in seconds from the beginning of content
watch-duration	How much time did the consumer watch the content, in seconds
watch-delay	Age of content in catch-up (difference between the time it was aired and the time it was watched), in seconds
content-duration	Content duration, in seconds
price	Price the consumer had to pay to be able to consume the content
origin	Origin of the event from within the user interface where the consumer triggered the event: either "recommendations" or "other"

interaction-type	For interaction type events: the type of interaction that the consumer did in connection with the content
feedback	Feedback description of the feedback-type event.
user-profile-id	The consumer's profile that the event was triggered from.

Note that if the system uses consumer profiles, then every event should specify the "user-profile-id" descriptor if possible.

The REST API at the path /rest/event is used to deliver events, with the HTTP method PUT.

Events that are described in the rest of this document can be triggered by a consumer using a digital receiver (set-top box), therefore it is required to supply with events among others the descriptors "device-id" (to pair events together) and "device-type" (to classify/categorize events).

Not all descriptors are sensible with every type of event. The possible (sensible) tuples of event type and product type are listed in the table below.

Event type	Product type	Descriptors
purchase	video	device-id, device-type, price, origin, catalogue-id
purchase	package	device-id, device-type, price, origin
consumption	video	device-id, device-type, watch-offset, watch-duration, content-duration, origin, catalogue-id, was-purchased
consumption	tv-channel	device-id, device-type, watch-offset, watch-duration, origin
consumption	tv-programme	device-id, device-type, watch-offset, watch-duration, content-duration, origin, watch-delay, tv-channel-id
zap	tv-channel	device-id, device-type, watch-offset, content-duration, tv-programme-id
zap	tv-programme	device-id, device-type, watch-offset, content-duration, action, speed, tv-channel-id
zap	video	device-id, device-type, watch-offset, content-duration, action, speed
rating	video, tv-programme	device-id, device-type, rating, origin
interaction	video, tv-programme	device-id, device-type, interaction-type, origin

5.1 Purchase

Purchasing habits of a consumer are a significant information for the recommendation system. A purchase event should be sent at every purchase.

5.1.1 TVOD

Before a consumer can watch a chargeable movie (TVOD) he has to buy it. After the purchase is completed a purchase event is generated. Every watching of the movie is signalled with a separate consumption event. Below is an example of a purchase event request in XML.

<request>

```
  <type>purchase</type>
  <user-id>81ec5b8f1c8bfa3d7f4dc3871e11d91c</user-id>
  <product-type>video</product-type>
  <product-id>video-test</product-id>
  <timestamp>1372523977</timestamp>
  <data>
    <item>
      <identifier>catalogue-id</identifier>
      <value>2</value>
    </item>
    <item>
      <identifier>device-id</identifier>
      <value>43e22fd8d493a1358d9f2e736f42babf</value>
    </item>
    <item>
      <identifier>device-type</identifier>
      <value>stbA</value>
    </item>
    <item>
      <identifier>price</identifier>
      <value>3.00</value>
    </item>
    <item>
      <identifier>origin</identifier>
      <value>recommendations</value>
    </item>
  </data>
</request>
```

Below is the same request in JSON.

```
{
  "type": "purchase",
  "user-id": "81ec5b8f1c8bfa3d7f4dc3871e11d91c",
  "product-type": "video",
  "product-id": "video-test",
  "timestamp": 1372523977,
  "data": [
    {
      "identifier": "catalogue-id",
      "value": "2"
    },
    {
      "identifier": "device-id",
      "value": "43e22fd8d493a1358d9f2e736f42babf"
    },
    {
      "identifier": "device-type",
      "value": "stbA"
    },
    {
      "identifier": "price",
      "value": 3.00
    },
    {
      "identifier": "origin",
      "value": "recommendations"
    }
  ]
}
```

5.1.2 Packages

Below is an example in XML for an event request that describes a package (subscription) purchase.

```
<request>
  <type>purchase</type>
  <user-id>81ec5b8f1c8bfa3d7f4dc3871e11d91c</user-id>
  <product-type>package</product-type>
  <product-id>package-test</product-id>
  <timestamp>1372523621</timestamp>
  <data>
    <item>
      <identifier>device-id</identifier>
      <value>43e22fd8d493a1358d9f2e736f42babf</value>
    </item>
    <item>
      <identifier>device-type</identifier>
      <value>stbA</value>
    </item>
    <item>
      <identifier>price</identifier>
      <value>7.00</value>
    </item>
    <item>
      <identifier>origin</identifier>
      <value>recommendations</value>
    </item>
  </data>
</request>
```

Below is the same request in JSON.

```
{
  "type": "purchase",
  "user-id": "81ec5b8f1c8bfa3d7f4dc3871e11d91c",
  "product-type": "package",
  "product-id": "package-test",
  "timestamp": "1372523621",
  "data": [
    {
      "identifier": "device-id",
      "value": "43e22fd8d493a1358d9f2e736f42babf"
    },
    {
      "identifier": "device-type",
      "value": "stbA"
    },
    {
      "identifier": "price",
      "value": 7.00
    },
    {
      "identifier": "origin",
      "value": "recommendations"
    }
  ]
}
```

5.2 Consumption

The consumption event type is used when a consumer watched some content (e.g. a movie). There are 3 different ways to watch a content, therefore there are 3 different event requests that can be generated.

Consumption events can be generated by the recommendation system from zap events if so configured.

5.2.1 SVOD/TVOD Consumption

When a consumer finishes watching a movie a "consumption" event is sent with the information about what and how the consumer watched the movie.

Below is an example of a request in XML.

```
<request>
  <type>consumption</type>
  <user-id>81ec5b8f1c8bfa3d7f4dc3871e11d91c</user-id>
  <product-type>video</product-type>
  <product-id>video-test</product-id>
  <timestamp>1372524834</timestamp>
  <data>
    <item>
      <identifier>catalogue-id</identifier>
      <value>2</value>
    </item>
    <item>
      <identifier>device-id</identifier>
      <value>43e22fd8d493a1358d9f2e736f42babf</value>
    </item>
    <item>
      <identifier>device-type</identifier>
      <value>stbA</value>
    </item>
    <item>
      <identifier>was-purchased</identifier>
      <value>false</value>
    </item>
    <item>
      <identifier>watch-offset</identifier>
      <value>0</value>
    </item>
    <item>
      <identifier>watch-duration</identifier>
      <value>857</value>
    </item>
    <item>
      <identifier>content-duration</identifier>
      <value>7920</value>
    </item>
    <item>
      <identifier>origin</identifier>
      <value>recommendations</value>
    </item>
  </data>
</request>
```

Below is the same request in JSON.


```
{
  "type": "consumption",
  "user-id": "81ec5b8f1c8bfa3d7f4dc3871e11d91c",
  "product-type": "video",
  "product-id": "video-test",
  "timestamp": 1372524834,
  "data": [
    {
      "identifier": "catalogue-id",
      "value": "2"
    },
    {
      "identifier": "device-id",
      "value": "43e22fd8d493a1358d9f2e736f42babf"
    },
    {
      "identifier": "device-type",
      "value": "stbA"
    },
    {
      "identifier": "was-purchased",
      "value": "false"
    },
    {
      "identifier": "watch-offset",
      "value": 0
    },
    {
      "identifier": "watch-duration",
      "value": 857
    },
    {
      "identifier": "content-duration",
      "value": 7920
    },
    {
      "identifier": "origin",
      "value": "recommendations"
    }
  ]
}
```


5.2.2 Live-TV Consumption

A "live-tv-consumption" event is sent in two cases:

- when a consumer switches between two TV-channels, an event is generated that refers to the TV-show running on the previously active TV-channel at the time of the switch,
- when a TV-show ends on the currently active TV-channel, an event is generated that refers to the TV-show that ended.

If the TV-channel in question has no EPG, then no event is generated, as there is no relevant information available.

Below is an example of a "live-tv-consumption" event request in XML.


```
<request>
```

```
  <type>live-tv-consumption</type>
  <user-id>81ec5b8f1c8bfa3d7f4dc3871e11d91c</user-id>
  <product-type>tv-programme</product-type>
  <product-id>468562576</product-id>
  <timestamp>1372524834</timestamp>
  <data>
    <item>
      <identifier>device-id</identifier>
      <value>43e22fd8d493a1358d9f2e736f42babf</value>
    </item>
    <item>
      <identifier>device-type</identifier>
      <value>stbA</value>
    </item>
    <item>
      <identifier>watch-offset</identifier>
      <value>274</value>
    </item>
    <item>
      <identifier>watch-duration</identifier>
      <value>21</value>
    </item>
    <item>
      <identifier>content-duration</identifier>
      <value>1800</value>
    </item>
    <item>
      <identifier>origin</identifier>
      <value>other</value>
    </item>
    <item>
      <identifier>tv-channel-id</identifier>
      <value>tv-channel-test</value>
    </item>
  </data>
</request>
```


Below is the same request in JSON.


```
{
  "type": "live-tv-consumption",
  "user-id": "81ec5b8f1c8bfa3d7f4dc3871e11d91c",
  "product-type": "tv-programme",
  "product-id": "468562576",
  "timestamp": 1372524834,
  "data": [
    {
      "identifier": "device-id",
      "value": "43e22fd8d493a1358d9f2e736f42babf"
    },
    {
      "identifier": "device-type",
      "value": "stbA"
    },
    {
      "identifier": "watch-offset",
      "value": 274
    },
    {
      "identifier": "watch-duration",
      "value": 21
```

```
    },
    {
      "identifier": "content-duration",
      "value": 1800
    },
    {
      "identifier": "origin",
      "value": "other"
    },
    {
      "identifier": "tv-channel-id",
      "value": "tv-channel-test"
    }
  ]
}
```

5.2.3 Catch-up Consumption

When a consumer is finished watching a TV-show from catch-up, a "consumption" event similar to a "live-tv-consumption" event is sent. The difference is in the event type and there is the additional descriptor "watch-delay".

Below is an example of a catch-up consumption request in XML.

```
<request>
  <type>consumption</type>
  <user-id>81ec5b8f1c8bfa3d7f4dc3871e11d91c</user-id>
  <product-type>tv-programme</product-type>
  <product-id>tv-programme-test</product-id>
  <timestamp>1372524834</timestamp>
  <data>
    <item>
      <identifier>device-id</identifier>
      <value>43e22fd8d493a1358d9f2e736f42babf</value>
    </item>
    <item>
      <identifier>device-type</identifier>
      <value>stbA</value>
    </item>
    <item>
      <identifier>watch-delay</identifier>
      <value>25239</value>
    </item>
    <item>
      <identifier>watch-offset</identifier>
      <value>0</value>
    </item>
    <item>
      <identifier>watch-duration</identifier>
      <value>327</value>
    </item>
    <item>
      <identifier>content-duration</identifier>
      <value>1800</value>
    </item>
    <item>
      <identifier>origin</identifier>
      <value>other</value>
    </item>
    <item>
```

```
      <identifier>tv-channel-id</identifier>
      <value>tv-channel-test</value>
    </item>
  </data>
</request>
```

Below is the same request in JSON.

```
{
  "type": "consumption",
  "user-id": "81ec5b8f1c8bfa3d7f4dc3871e11d91c",
  "product-type": "tv-programme",
  "product-id": "tv-programme-test",
  "timestamp": 1372524834,
  "data": [
    {
      "identifier": "device-id",
      "value": "43e22fd8d493a1358d9f2e736f42babf"
    },
    {
      "identifier": "device-type",
      "value": "stbA"
    },
    {
      "identifier": "watch-delay",
      "value": 25239
    },
    {
      "identifier": "watch-offset",
      "value": 0
    },
    {
      "identifier": "watch-duration",
      "value": 327
    },
    {
      "identifier": "content-duration",
      "value": 1800
    },
    {
      "identifier": "origin",
      "value": "other"
    },
    {
      "identifier": "tv-channel-id",
      "value": "tv-channel-test"
    }
  ]
}
```

5.3 TV-channel Switching, Playing and Device Control (Zaps)

The recommendation system requires information about the current viewing of TV-channels. This information is computed from TV-channel zaps, which are basically events describing a switch to a new TV-channel on a set-top box. As an added feature, the recommendation system can take over generating "live-tv-consumption" events, based on TV-channel zaps, if so configured. In this case it is essential that there is no zap event loss and that the information given in zap events is accurate. Besides TV-channel zaps there are also possible playing control zaps, to describe playing status

(play, fast-forward, rewind, pause, stop) of a video or catch-up content, as a basis for computing catch-up and VOD consumptions. Device control zaps are a special category of zaps that don't refer to a product, but instead describe an aspect of the device operation.

5.3.1 TV-channel Zaps

A TV-channel zap is a zap event the refers to a product of the "tv-channel" type and describes a switch to a new TV-channel.

Below is an example of a TV-channel zap request in XML.

```
<request>
  <type>zap</type>
  <user-id>81ec5b8f1c8bfa3d7f4dc3871e11d91c</user-id>
  <product-type>tv-channel</product-type>
  <product-id>tv-channel-test</product-id>
  <timestamp>1372524892</timestamp>
  <data>
    <item>
      <identifier>device-id</identifier>
      <value>43e22fd8d493a1358d9f2e736f42babf</value>
    </item>
    <item>
      <identifier>device-type</identifier>
      <value>stbA</value>
    </item>
    <item>
      <identifier>tv-programme-id</identifier>
      <value>344576567</value>
    </item>
    <item>
      <identifier>watch-offset</identifier>
      <value>326</value>
    </item>
    <item>
      <identifier>content-duration</identifier>
      <value>1800</value>
    </item>
    <item>
      <identifier>origin</identifier>
      <value>other</value>
    </item>
  </data>
</request>
```

Below is the same request in JSON.

```
{
  "type": "zap",
  "user-id": "81ec5b8f1c8bfa3d7f4dc3871e11d91c",
  "product-type": "tv-channel",
  "product-id": "tv-channel-test",
  "timestamp": 1372524892,
  "data": [
    {
      "identifier": "device-id",
      "value": "43e22fd8d493a1358d9f2e736f42babf"
    },
    {
```

```
        "identifier": "device-type",
        "value": "stbA"
      },
      {
        "identifier": "tv-programme-id",
        "value": "344576567"
      },
      {
        "identifier": "watch-offset",
        "value": 326
      },
      {
        "identifier": "content-duration",
        "value": 1800
      },
      {
        "identifier": "origin",
        "value": "recommendations"
      }
    ]
}
```


5.3.2 Playing Control Zaps

Playing control zaps describe changes in the playing of a VOD or catch-up content, which are usually generated by keypresses on a TV-remote control on play, rewind, fast-forward, pause and stop buttons. Usually there is picture shown not only during play, but also during rewind and fast-forward winding, so these three states are reduced to one, having an additional information: speed. Speed can be 1 or missing for play, greater than 1 for fast-forward, and negative for rewind. This essentially mirrors the RTSP protocol semantics. For catch-up zaps the referenced product must be of the "tv-programme" type, and for VOD zaps the referenced product must be of the "video" type. Possible descriptors are: "device-id", "device-type", "origin", "action", "speed", "watch-offset", and only with "tv-programme" product type: "tv-channel-id".

Playing control zaps are needed only if the recommendation system is to generate VOD and catch-up consumption events.

Below is an example of a playing control zap request in XML, referencing a catch-up content.


```
<request>
  <type>zap</type>
  <user-id>81ec5b8f1c8bfa3d7f4dc3871e11d91c</user-id>
  <product-type>tv-programme</product-type>
  <product-id>14363295</product-id>
  <timestamp>1412785679</timestamp>
  <data>
    <item>
      <identifier>device-id</identifier>
      <value>43e22fd8d493a1358d9f2e736f42babf</value>
    </item>
    <item>
      <identifier>device-type</identifier>
      <value>stbA</value>
    </item>
    <item>
      <identifier>origin</identifier>
```

```
      <value>recommendations</value>
    </item>
    <item>
      <identifier>tv-channel-id</identifier>
      <value>tv-channel-test</value>
    </item>
    <item>
      <identifier>action</identifier>
      <value>play</value>
    </item>
    <item>
      <identifier>watch-offset</identifier>
      <value>326</value>
    </item>
  </data>
</request>
```


Below is the same request in JSON.


```
{
  "type": "zap",
  "user-id": "81ec5b8f1c8bfa3d7f4dc3871e11d91c",
  "product-type": "tv-programme",
  "product-id": "14363295",
  "timestamp": 1412785679,
  "data": [
    {
      "identifier": "device-id",
      "value": "43e22fd8d493a1358d9f2e736f42babf"
    },
    {
      "identifier": "device-type",
      "value": "stbA"
    },
    {
      "identifier": "origin",
      "value": "recommendations"
    },
    {
      "identifier": "tv-channel-id",
      "value": " tv-channel-test"
    },
    {
      "identifier": "action",
      "value": "play"
    },
    {
      "identifier": "watch-offset",
      "value": 326
    }
  ]
}
```


5.3.3 Device Control Zaps

When there is a change in a device operation a special zap is sent that does not refer to (does not include data about) a product. Currently only two such aspects are of interest to the recommendation system: when the device is turned on and when the device is turned off. These

two states are described using the "status" descriptor, which can be either "power-on" or "power-off".

Below is an example of a "power-off" zap event request in XML.

```
<request>
  <type>zap</type>
  <user-id>81ec5b8f1c8bfa3d7f4dc3871e11d91c</user-id>
  <timestamp>1412785679</timestamp>
  <data>
    <item>
      <identifier>device-id</identifier>
      <value>43e22fd8d493a1358d9f2e736f42babf</value>
    </item>
    <item>
      <identifier>device-type</identifier>
      <value>stbA</value>
    </item>
    <item>
      <identifier>status</identifier>
      <value>power-off</value>
    </item>
  </data>
</request>
```

Below is the same request in JSON.

```
{
  "type": "zap",
  "user-id": "81ec5b8f1c8bfa3d7f4dc3871e11d91c",
  "timestamp": 1412785679,
  "data": [
    {
      "identifier": "device-id",
      "value": "43e22fd8d493a1358d9f2e736f42babf"
    },
    {
      "identifier": "device-type",
      "value": "stbA"
    },
    {
      "identifier": "status",
      "value": "power-off"
    }
  ]
}
```

5.4 Content Rating

A rating event is sent when a consumer rates a content. The rating itself is included as the "rating" descriptor. It is only sensible to rate VOD and EPG content, therefore the only possible product types for the rating event are "video" and "tv-programme".

Below is an example of a rating event request in XML.

```
<request>
  <type>rating</type>
  <user-id>81ec5b8f1c8bfa3d7f4dc3871e11d91c</user-id>
  <product-type>video</product-type>
  <product-id>video-test</product-id>
```

```
  <timestamp>1372524867</timestamp>
  <data>
    <item>
      <identifier>rating</identifier>
      <value>3</value>
    </item>
    <item>
      <identifier>device-id</identifier>
      <value>43e22fd8d493a1358d9f2e736f42babf</value>
    </item>
    <item>
      <identifier>device-type</identifier>
      <value>stbA</value>
    </item>
    <item>
      <identifier>origin</identifier>
      <value>recommendations</value>
    </item>
  </data>
</request>
```

Below is the same request in JSON.

```
{
  "type": "rating",
  "user-id": "81ec5b8f1c8bfa3d7f4dc3871e11d91c",
  "product-type": "video",
  "product-id": "video-test",
  "timestamp": 1372524867,
  "data": [
    {
      "identifier": "rating",
      "value": 3
    },
    {
      "identifier": "device-id",
      "value": "43e22fd8d493a1358d9f2e736f42babf"
    },
    {
      "identifier": "device-type",
      "value": "stbA"
    },
    {
      "identifier": "origin",
      "value": "recommendations"
    }
  ]
}
```

5.5 User Experience Monitoring

In certain cases the user interaction is an important source for determining a consumer's taste.

5.5.1 Viewing a Content Description

As a general rule a consumer requests a content description when the consumer becomes interested in the content. Interest shown in this way is an important piece of information for the recommendation system.

Below is an example of an event request describing a consumer viewing a content description, in XML.

```
<request>
  <type>interaction</type>
  <user-id>81ec5b8f1c8bfa3d7f4dc3871e11d91c</user-id>
  <product-type>video</product-type>
  <product-id>video-test</product-id>
  <timestamp>1372524834</timestamp>
  <data>
    <item>
      <identifier>catalogue-id</identifier>
      <value>2</value>
    </item>
    <item>
      <identifier>device-id</identifier>
      <value>43e22fd8d493a1358d9f2e736f42babf</value>
    </item>
    <item>
      <identifier>device-type</identifier>
      <value>stbA</value>
    </item>
    <item>
      <identifier>interaction-type</identifier>
      <value>content-description</value>
    </item>
    <item>
      <identifier>origin</identifier>
      <value>recommendations</value>
    </item>
  </data>
</request>
```

Below is the same request in JSON.

```
{
  "type": "interaction",
  "user-id": "81ec5b8f1c8bfa3d7f4dc3871e11d91c",
  "product-type": "video",
  "product-id": "video-test",
  "timestamp": 1372524834,
  "data": [
    {
      "identifier": "catalogue-id",
      "value": "2"
    },
    {
      "identifier": "device-id",
      "value": "43e22fd8d493a1358d9f2e736f42babf"
    },
    {
      "identifier": "device-type",
      "value": "stbA"
    },
    {
      "identifier": "interaction-type",
      "value": "content-description"
    },
    {
      "identifier": "origin",
      "value": "recommendations"
```

```
    }
  ]
}
```

5.6 Content feedback

User can provide an explicit feedback about a certain content. Most often this is the case when the consumer wishes to add content among his favorites, or block content from appearing among recommendations in the future. The feedback is taken into account when recommendations are generated for the consumer. It is possible to extend the range of possible feedback with additional types of feedback.

The feedback is sent using an event of type "feedback" with the description of the feedback in the data descriptor "feedback". Out of the box the system supports the "favorite", "unfavorite", "block", and "unblock" feedbacks.

Below is an example of an event request specifying that a consumer favorited a certain content, in XML.

```
<request>
  <type>feedback</type>
  <user-id>81ec5b8f1c8bfa3d7f4dc3871e11d91c</user-id>
  <product-type>video</product-type>
  <product-id>video-test</product-id>
  <timestamp>1372524834</timestamp>
  <data>
    <item>
      <identifier>feedback</identifier>
      <value>favorite</value>
    </item>
    <item>
      <identifier>device-id</identifier>
      <value>43e22fd8d493a1358d9f2e736f42babf</value>
    </item>
    <item>
      <identifier>device-type</identifier>
      <value>stbA</value>
    </item>
    <item>
      <identifier>origin</identifier>
      <value>recommendations</value>
    </item>
  </data>
</request>
```

Below is the same request in JSON.

```
{
  "type": "feedback",
  "user-id": "81ec5b8f1c8bfa3d7f4dc3871e11d91c",
  "product-type": "video",
  "product-id": "video-test",
  "timestamp": 1372524834,
  "data": [
    {
      "identifier": "feedback",
      "value": "favorite"
```

```
    },
    {
      "identifier": "device-id",
      "value": "43e22fd8d493a1358d9f2e736f42babf"
    },
    {
      "identifier": "device-type",
      "value": "stbA"
    },
    {
      "identifier": "origin",
      "value": "recommendations"
    }
  ]
}
```

6 Recommendations

The recommendations service is the central part of the recommendation system, it returns a list of content chosen based on the given parameters.

Behind the recommendation service there are many recommendation engines tuned to specifics of their domains, that specialize in recommending certain types of content. For example, there can be an engine for movies, an engine for tv-series, or a general engine for mixed content. Specifying a recommendation type (recommender for short) selects a certain recommendation engine, with some additional content filtering applied, so in general an engine can be used with more than one recommender. We strive to define recommenders in such a way that they reflect all the possible choices in the partner's application, so in general it is sufficient to supply a consumer ID and a recommendation type with a recommendation request.

Each recommender is configured with a default maximum number of recommendation items to return. This can be overridden with the "maxRecommendations" request element.

Usually all the content pertinent to the selected recommender is considered while creating recommendations. This content can be filtered before the recommendations are created by including a list of attributes with the recommendation request. Then only the content that contains the provided attributes is considered for creating recommendations. Usually the categorizing attributes, such as genre, are used.

A recommendation request can contain a list of event descriptors, semantically this is the same list as the event descriptors in event requests. The descriptors are needed to describe the request in a greater detail, at a minimum with the data about the device it was sent from ("device-id", "device-type"). This data is used by the recommendation system and analytics.

Note that if the system uses consumer profiles, then every request should also specify the "user-profile-id" event descriptor if possible.

For recommendations the REST API at the path /rest/recommendations is used, with the HTTP method POST. Below is the list of possible elements that can be contained in a recommendation request.

Element	**Description**	**Mandatory**
user-id	Consumer for which to generate recommendations	yes
type	Recommendation type	yes
data	A list of event descriptors	no
	“user-profile-id" event descriptor	when consumer profiles in use
product	The type and ID of a product for which to find similar content	when "type"="similar"
maxRecommendations	Maximum number of items to return	no

attributes	A list of attributes that the returned items must contain	no
instance	The result set instance to use, defaults to "new"	no

The service replies with a response that contains recommended products (listed as product type and product ID) and if so configured also certain attributes for certain product types. The recommendation response contains "resultCode" and "resultMessage", which are contained in every response of recommendation system services to indicate a success or failure processing the request, and "recommendations" which is a list of recommended content and present only when "responseCode" represents a success.

If so desired the response can be configured to contain explanations for every content in the response why it was recommended.

Below is an example of a recommendation response in XML.

```
<response>
  <resultCode>0</resultCode>
  <resultMessage>OK</resultMessage>
  <resultCount>3</resultCount>
  <recommendations>
    <recommendation>
      <type>video</type>
      <id>123</id>
      <explanations>
        <explanation>you like: genre:adventure</explanation>
        <explanation>Similar users liked this item</explanation>
      </explanations>
    </recommendation>
    <recommendation>
      <type>tv-programme</type>
      <id>246</id>
      <attributes>
        <attribute>
          <identifier>tv-channel</identifier>
          <value>CNN</value>
        </attribute>
        <attribute>
          <identifier>begin-time</identifier>
          <value>1379946900</value>
        </attribute>
        <attribute>
          <identifier>end-time</identifier>
          <value>1379947200</value>
        </attribute>
      </attributes>
      <explanations>
        <explanation>Similar users liked this item</explanation>
        <explanation>Users who watched title:CNN Money also watched this item</explanation>
      </explanations>
    </recommendation>
    <recommendation>
      <type>video</type>
      <id>789</id>
      <explanations>
        <explanation>Currently watched series</explanation>
        <explanation>Users who watched title:Cube also watched this item</explanation>
```

```
            <explanation>you like: genre:sci-fi</explanation>
          </explanations>
        </recommendation>
      </recommendations>
    </response>
```

Below is the same response in JSON.


```
{
  "resultCode": 0,
  "resultMessage": "OK",
  "resultCount": 3,
  "recommendations": [
    {
      "type": "video",
      "id": "123",
      "explanations": [
        {
          "value": "you like: genre:adventure"
        },
        {
          "value": "Similar users liked this item"
        }
      ]
    },
    {
      "type": "tv-programme",
      "id": "246",
      "attributes": [
        {
          "identifier": "tv-channel",
          "value": "CNN"
        },
        {
          "identifier": "begin-time",
          "value": "1379946900"
        },
        {
          "identifier": "end-time",
          "value": "1379947200"
        }
      ],
      "explanations": [
        {
          "value": "Similar users liked this item"
        },
        {
          "value": "Users who watched title:CNN Money also watched this item"
        }
      ]
    },
    {
      "type": "video",
      "id": "789",
      "explanations": [
        {
          "value": "Currently watched series"
        },
        {
          "value": "Users who watched title:Cube also watched this item"
        },
```

```
        {
          "value": "you like: genre:sci-fi"
        }
      ]
    }
  ]
}
```

Following are examples of predefined recommenders.

6.1 General Recommendations (all)

General recommendations result in a list of recommendations of mixed content as all the available content is considered, therefore this recommendation type is named "all". The response may contain live-TV, catch-up and VOD content, represented with "video" and "tv-programme" products.

Below is an example of a request for general recommendations in XML.

```
<request>
  <user-id>81ec5b8f1c8bfa3d7f4dc3871e11d91c</user-id>
  <type>all</type>
  <data>
    <item>
      <identifier>device-id</identifier>
      <value>43e22fd8d493a1358d9f2e736f42babf</value>
    </item>
    <item>
      <identifier>device-type</identifier>
      <value>stbA</value>
    </item>
    <item>
      <identifier>user-profile-id</identifier>
      <value>James</value>
    </item>
  </data>
</request>
```

Below is the same request in JSON.

```
{
  "user-id": "81ec5b8f1c8bfa3d7f4dc3871e11d91c",
  "type": "all",
  "data": [
    {
      "identifier": "device-id",
      "value": "43e22fd8d493a1358d9f2e736f42babf"
    },
    {
      "identifier": "device-type",
      "value": "stbA"
    },
    {
      "identifier": "user-profile-id",
      "value": "James"
    }
  ]
}
```

6.2 Live-TV (tv)

TV recommender returns a list of "tv-programme" recommendations, that is catch-up and live-TV content. Both content categories are represented with products of "tv-programme" type, the difference is that catch-up content was already aired in the past but is still available through partner's playout servers, while live-TV content is airing at the moment (in real-time), or is scheduled to be aired in the near future.

Below is an example of a request for TV recommendations in XML.

```
<request>
  <user-id>81ec5b8f1c8bfa3d7f4dc3871e11d91c</user-id>
  <type>tv</type>
  <data>
    <item>
      <identifier>device-id</identifier>
      <value>43e22fd8d493a1358d9f2e736f42babf</value>
    </item>
    <item>
      <identifier>device-type</identifier>
      <value>stbA</value>
    </item>
  </data>
</request>
```

Below is the same request in JSON.

```
{
  "user-id": "81ec5b8f1c8bfa3d7f4dc3871e11d91c",
  "type": "tv",
  "data": [
    {
      "identifier": "device-id",
      "value": "43e22fd8d493a1358d9f2e736f42babf"
    },
    {
      "identifier": "device-type",
      "value": "stbA"
    }
  ]
}
```

6.3 TV-series (tv-series)

To retrieve recommendations only for TV-series, the "tv-series" recommender is used. Returned is a list of mixed content (VOD, catch-up, live-TV), where each recommended content represents an episode of a TV-series.

Below is an example of a request for TV-series recommendations in XML.

```
<request>
  <user-id>81ec5b8f1c8bfa3d7f4dc3871e11d91c</user-id>
  <type>tv-series</type>
  <data>
    <item>
      <identifier>device-id</identifier>
      <value>43e22fd8d493a1358d9f2e736f42babf</value>
    </item>
```

```
    <item>
      <identifier>device-type</identifier>
      <value>stbA</value>
    </item>
  </data>
</request>
```

Below is the same request in JSON.

```
{
  "user-id": "81ec5b8f1c8bfa3d7f4dc3871e11d91c",
  "type": "tv-series",
  "data": [
    {
      "identifier": "device-id",
      "value": "43e22fd8d493a1358d9f2e736f42babf"
    },
    {
      "identifier": "device-type",
      "value": "stbA"
    }
  ]
}
```

6.4 Movies (movies)

To retrieve recommendations only for movies, the "movies" recommender is used. Returned is a list of mixed content (VOD, catch-up, live-TV), where each recommended content represents a movie.

Below is an example of a request for movie recommendations in XML.

```
<request>
  <user-id>81ec5b8f1c8bfa3d7f4dc3871e11d91c</user-id>
  <type>movie</type>
  <data>
    <item>
      <identifier>device-id</identifier>
      <value>43e22fd8d493a1358d9f2e736f42babf</value>
    </item>
    <item>
      <identifier>device-type</identifier>
      <value>stbA</value>
    </item>
  </data>
</request>
```

Below is the same request in JSON.

```
{
  "user-id": "81ec5b8f1c8bfa3d7f4dc3871e11d91c",
  "type": "movie",
  "data": [
    {
      "identifier": "device-id",
      "value": "43e22fd8d493a1358d9f2e736f42babf"
    },
    {
      "identifier": "device-type",
      "value": "stbA"
    }
```

```
  ]
}
```

6.5 Most Viewed Content (most-viewed)

To retrieve a list of most-viewed content, the "most-viewed" recommender is used. Returned is a list of mixed content (VOD, catch-up, live-TV). The time interval for determining the most-viewed content is configurable.

Below is an example of a request for movie recommendations in XML.

```
<request>
  <user-id>81ec5b8f1c8bfa3d7f4dc3871e11d91c</user-id>
  <type>most-viewed</type>
  <data>
    <item>
      <identifier>device-id</identifier>
      <value>43e22fd8d493a1358d9f2e736f42babf</value>
    </item>
    <item>
      <identifier>device-type</identifier>
      <value>stbA</value>
    </item>
  </data>
</request>
```

Below is the same request in JSON.

```
{
  "user-id": "81ec5b8f1c8bfa3d7f4dc3871e11d91c",
  "type": "most-viewed",
  "data": [
    {
      "identifier": "device-id",
      "value": "43e22fd8d493a1358d9f2e736f42babf"
    },
    {
      "identifier": "device-type",
      "value": "stbA"
    }
  ]
}
```

6.6 Similar Content (similar)

It is possible to retrieve a list of content that is similar to the given content. For this the "similar" recommender is used, and the request must also contain the "product" element with the "type" and "id" sub-elements containing the product type and the product ID. Returned is a list of mixed content (VOD, catch-up, live-TV) representing content that is similar to the given content.

Below is an example of a request for movie recommendations in XML.

```
<request>
  <user-id>81ec5b8f1c8bfa3d7f4dc3871e11d91c</user-id>
  <type>similar</type>
  <product>
    <type>video</type>
    <id>123</id>
  </product>
```

```
  <data>
    <item>
      <identifier>device-id</identifier>
      <value>43e22fd8d493a1358d9f2e736f42babf</value>
    </item>
    <item>
      <identifier>device-type</identifier>
      <value>stbA</value>
    </item>
  </data>
</request>
```

Below is the same request in JSON.

```
{
  "user-id": "81ec5b8f1c8bfa3d7f4dc3871e11d91c",
  "type": "similar",
  "product": {
    "type": "video",
    "id": "123"
  },
  "data": [
    {
      "identifier": "device-id",
      "value": "43e22fd8d493a1358d9f2e736f42babf"
    },
    {
      "identifier": "device-type",
      "value": "stbA"
    }
  ]
}
```

6.7 Live-TV Grid (tv-grid)

If the partner's GUI permits a grid-like representation of catch-up and live-TV content, with one dimension representing the timeline and the other dimension representing TV-channels, it is possible to retrieve EPG (catch-up and live-TV) recommendations for grid-like representation from the recommender "tv-grid". The time dimension represents the actual timeline spanning from the oldest available content in catch-up to some predefined time in the future (e.g. 15 minutes). The TV-channel dimension consists of a preconfigured fixed number of virtual TV-channels, which have no special meaning by default, but can be configured to contain only content of a certain genre, category, etc.

The response contains together with the data about each recommended content additionally the "gridLine" element. The "gridLine" value is the index of the virtual TV-channel where to position the content inside the grid. The position on the time dimension is defined with the "begin-time" and "end-time" attributes of the TV-content. The "tv-grid" recommender ensures that no two recommended items on the same virtual TV-channel intersect.

Below is an example of a request for live-TV grid recommendations in XML.

```
<request>
  <user-id>81ec5b8f1c8bfa3d7f4dc3871e11d91c</user-id>
  <type>tv-grid</type>
```

```
  <data>
    <item>
      <identifier>device-id</identifier>
      <value>43e22fd8d493a1358d9f2e736f42babf</value>
    </item>
    <item>
      <identifier>device-type</identifier>
      <value>stbA</value>
    </item>
  </data>
</request>
```

Below is the same request in JSON.

```
{
  "user-id": "81ec5b8f1c8bfa3d7f4dc3871e11d91c",
  "type": "tv-grid",
  "data": [
    {
      "identifier": "device-id",
      "value": "43e22fd8d493a1358d9f2e736f42babf"
    },
    {
      "identifier": "device-type",
      "value": "stbA"
    }
  ]
}
```

Below is an example of a "tv-grid" response in XML.

```
<response>
  <resultCode>0</resultCode>
  <resultMessage>OK</resultMessage>
  <recommendations>
    <recommendation>
      <type>tv-programme</type>
      <id>10533007</id>
      <gridLine>1</gridLine>
      <attributes>
        <attribute>
          <identifier>tv-channel</identifier>
          <value>CNN</value>
        </attribute>
        <attribute>
          <identifier>begin-time</identifier>
          <value>1380031200</value>
        </attribute>
        <attribute>
          <identifier>end-time</identifier>
          <value>1380034200</value>
        </attribute>
      </attributes>
    </recommendation>
    <recommendation>
      <type>tv-programme</type>
      <id>10532966</id>
      <gridLine>2</gridLine>
      <attributes>
        <attribute>
          <identifier>tv-channel</identifier>
          <value>ARD</value>
```

```
      </attribute>
      <attribute>
        <identifier>begin-time</identifier>
        <value>1379988000</value>
      </attribute>
      <attribute>
        <identifier>end-time</identifier>
        <value>1379991600</value>
      </attribute>
    </attributes>
  </recommendation>
  <recommendation>
    <type>tv-programme</type>
    <id>10624241</id>
    <gridLine>1</gridLine>
    <attributes>
      <attribute>
        <identifier>tv-channel</identifier>
        <value>MTV</value>
      </attribute>
      <attribute>
        <identifier>begin-time</identifier>
        <value>1380036900</value>
      </attribute>
      <attribute>
        <identifier>end-time</identifier>
        <value>1380038400</value>
      </attribute>
    </attributes>
  </recommendation>
  <recommendation>
    <type>tv-programme</type>
    <id>10532121</id>
    <gridLine>3</gridLine>
    <attributes>
      <attribute>
        <identifier>tv-channel</identifier>
        <value>RTL</value>
      </attribute>
      <attribute>
        <identifier>begin-time</identifier>
        <value>1380033000</value>
      </attribute>
      <attribute>
        <identifier>end-time</identifier>
        <value>1380033900</value>
      </attribute>
    </attributes>
  </recommendation>
  <recommendation>
    <type>tv-programme</type>
    <id>10531738</id>
    <gridLine>4</gridLine>
    <attributes>
      <attribute>
        <identifier>tv-channel</identifier>
        <value>VH1</value>
      </attribute>
      <attribute>
        <identifier>begin-time</identifier>
```

```
            <value>1379993400</value>
          </attribute>
          <attribute>
            <identifier>end-time</identifier>
            <value>1379998500</value>
          </attribute>
        </attributes>
      </recommendation>
    </recommendations>
</response>
```

Below is the same response in JSON.

```
{
  "resultCode": 0,
  "resultMessage": "OK",
  "recommendations": [
    {
      "type": "tv-programme",
      "id": "10533007",
      "gridLine": 1,
      "attributes": [
        {
          "identifier": "tv-channel",
          "value": "CNN"
        },
        {
          "identifier": "begin-time",
          "value": "1380031200"
        },
        {
          "identifier": "end-time",
          "value": "1380034200"
        }
      ]
    },
    {
      "type": "tv-programme",
      "id": "10532966",
      "gridLine": 2,
      "attributes": [
        {
          "identifier": "tv-channel",
          "value": "ARD"
        },
        {
          "identifier": "begin-time",
          "value": "1379988000"
        },
        {
          "identifier": "end-time",
          "value": "1379991600"
        }
      ]
    },
    {
      "type": "tv-programme",
      "id": "10624241",
      "gridLine": 1,
      "attributes": [
        {
```

```
            "identifier": "tv-channel",
            "value": "MTV"
          },
          {
            "identifier": "begin-time",
            "value": "1380036900"
          },
          {
            "identifier": "end-time",
            "value": "1380038400"
          }
        ]
      },
      {
        "type": "tv-programme",
        "id": "10532121",
        "gridLine": 3,
        "attributes": [
          {
            "identifier": "tv-channel",
            "value": "RTL"
          },
          {
            "identifier": "begin-time",
            "value": "1380033000"
          },
          {
            "identifier": "end-time",
            "value": "1380033900"
          }
        ]
      },
      {
        "type": "tv-programme",
        "id": "10531738",
        "gridLine": 4,
        "attributes": [
          {
            "identifier": "tv-channel",
            "value": "VH1"
          },
          {
            "identifier": "begin-time",
            "value": "1379993400"
          },
          {
            "identifier": "end-time",
            "value": "1379998500"
          }
        ]
      }
    ]
}
```


6.8 Filtering With Attributes

When a recommendation request includes a list of attributes, the response contains only content having those attributes. In general this is useful to restrict recommendations to a certain genre, actor, director, and similar.

Below is an example of a request for general recommendations, having the genre "documentary", in XML.

```
<request>
  <user-id>81ec5b8f1c8bfa3d7f4dc3871e11d91c</user-id>
  <type>all</type>
  <attributes>
    <attribute>
      <identifier>genre</identifier>
      <value>documentary</value>
    </attribute>
  </attributes>
  <data>
    <item>
      <identifier>device-id</identifier>
      <value>43e22fd8d493a1358d9f2e736f42babf</value>
    </item>
    <item>
      <identifier>device-type</identifier>
      <value>stbA</value>
    </item>
  </data>
</request>
```

Below is the same request in JSON.

```
{
  "user-id": "81ec5b8f1c8bfa3d7f4dc3871e11d91c",
  "type": "all",
  "attributes": [
    {
      "identifier": "genre",
      "value": "documentary"
    }
  ]
  "data": [
    {
      "identifier": "device-id",
      "value": "43e22fd8d493a1358d9f2e736f42babf"
    },
    {
      "identifier": "device-type",
      "value": "stbA"
    }
  ]
}
```

7 Search

The search service returns a list of content chosen based on the search query and given parameters.

Specifying a search type selects a certain search engine, with additional content filtering applied, so in general an engine can be used with more than one search type. This way search types can specialize in searching certain types of content.

Each search request is configured with a default maximum number of content to return. This can be overridden with the "maxItems" request element.

A search request can contain a list of event descriptors, semantically this is the same list as the event descriptors in event requests. The descriptors are needed to describe the request in a greater detail, at a minimum with the data about the device it was sent from ("device-id", "device-type"). This data is used by search engines and analytics.

For search service the REST API at the path /rest/search is used, with the HTTP method POST. Below is the list of possible elements that can be contained in a search request.

Element	Description	Mandatory
user-id	Consumer for which to generate search	yes
query	Search query	yes
type	Search type	no
data	A list of event descriptors	no
maxItems	Maximum number of items to return	no

The service replies with a response that contains content found (listed as product type and product ID) and if so configured also certain attributes for certain product types. The search response contains "resultCode", "resultMessage" and "resultSet" which is a list of content found and present only when "responseCode" represents a success.

Below is an example of search request in XML.

```
<request>
  <user-id>81ec5b8f1c8bfa3d7f4dc3871e11d91c</user-id>
  <query>fifth</query>
  <type>all</type>
  <data>
    <item>
      <identifier>device-id</identifier>
      <value>43e22fd8d493a1358d9f2e736f42babf</value>
    </item>
    <item>
      <identifier>device-type</identifier>
      <value>stbA</value>
    </item>
  </data>
</request>
```

Below is the same request in JSON.

```
{
```

```
  "user-id": "81ec5b8f1c8bfa3d7f4dc3871e11d91c",
  "query": "fifth",
  "type": "all",
  "data": [
    {
      "identifier": "device-id",
      "value": "43e22fd8d493a1358d9f2e736f42babf"
    },
    {
      "identifier": "device-type",
      "value": "stbA"
    }
  ]
}
```

Below is an example of a search response in XML.

```
<response>
  <resultCode>0</resultCode>
  <resultMessage>OK</resultMessage>
  <resultsSet>
    <result>
      <type>video</type>
      <id>123</id>
    </result>
    <result>
      <type>tv-programme</type>
      <id>246</id>
    </result>
    <result>
      <type>video</type>
      <id>789</id>
    </result>
  </ resultsSet >
</response>
```

Below is the same response in JSON.

```
{
  "resultCode": 0,
  "resultMessage": "OK",
  " resultsSet ": [
    {
      "type": "video",
      "id": "123"
    },
    {
      "type": "tv-programme",
      "id": "246",
    },
    {
      "type": "video",
      "id": "789"
    }
  ]
}
```

8 Health

The successful response from health service means RESTful web is operational, in the oposite something is out of order.

For health service the REST API at the path /rest/health is used, with the HTTP method GET.

Below is an example of a health response in XML.

```
<response>
   <resultCode>0</resultCode>
   <resultMessage>OK</resultMessage>
</response>
```

Below is the same response in JSON.

```
{
   "resultCode": 0,
   "resultMessage": "OK"
}
```