10 March 2020

EMA/567640/2018

Substance, Product, Organisation, Referentials (SPOR)

SPOR API v2 Specification

30 Churchill Place ● Canary Wharf ● London E14 5EU ● United Kingdom
Telephone +44 (0)20 3660 6000 **Facsimile** +44 (0)20 3660 5555
Send a question via our website www.ema.europa.eu/contact

Table of Contents

1. Document Purpose

The purpose of this document is to present the description of the intended SPOR Application Programming Interface (API) version 2.0.

This SPOR API 2.0 is based on the international FHIR standard (Fast Healthcare Interoperability Resources), http://hl7.org/fhir

2. Context

The requirements for these API specifications are originated in those of the Substance Management System and the Product Management System as identified by the different stakeholders who own and contribute to the SPOR programme.

The specifications for the SPOR API are a first step towards implementing the system supporting the API.

3. Scope

The scope is defined as the new SPOR API 2.0 only.

This specification is based on the latest version of FHIR , available at http://build.fhir.org/resourcelist.html, built on top of R4 (v4.0.1) and in interim state until R5 publication. See http://hl7.org/fhir/directory.html for a list of all FHIR versions.

4. Introduction

4.1. *FHIR Introduction*

FHIR is a recent standard from HL7 that makes it easy and quick to build REST based APIs for healthcare applications. FHIR solutions are built from a set of modular components called "Resources".

A general introduction to FHIR can be found here:

http://hl7.org/fhir/summary.html

Developers and architects may wish to read these more technical overviews:

http://hl7.org/fhir/overview-dev.html

http://hl7.org/fhir/overview-arch.html

Those with a clinical background can start here:

http://hl7.org/fhir/overview-clinical.html

In general, this specification will not copy or repeat information that is available in the FHIR standard (http://hl7.org/fhir). Instead, references to the relevant parts of FHIR are given. A general knowledge of FHIR, gained from reading the introductions above and further reading at http://hl7.org/fhir will be necessary to fully interpret this specification.

Although references to FHIR are given, FHIR is a wide and flexible system, and not every aspect of FHIR specification will be supported. This document shows which parts of FHIR do apply to SPOR API 2.0. The parts that are supported shall conform to the FHIR specification rules.

FHIR is an emerging standard and is being actively tested in many live implementations around the world. Therefore it is possible that changes to FHIR may requires changes to this specification, while it is draft (but not once published and finalized).

FHIR documentation references in this API refer to the current major release of FHIR called STU3 (at http://hl7.org/fhir) for general FHIR information that is not subject to change. References to the resources used in this API are given to a the build server draft version (http://build.fhir.org/resourcelist.html), because the resources have been updated since the last formal FHIR release (http://hl7.org/fhir/R4/resourcelist.html). Resources that pre-existed the SPOR API, are still referred to in their R4 version (for example http://hl7.org/fhir/R4/task.html). It is expected that FHIR will create some intermediate release in January 2020, which then will reflect a draft state of the resources before R5.

4.2. *Definitions*

An API can be defined in various ways; the definitions below form a good start for this[1]:

1. "It is a set of **routines**, **protocols**, and tools for building software applications."
2. "It expresses a **software component** in terms of its **operations**, **inputs**, **outputs**, and **underlying types**."

If we take the second definition we can expand on the terms used, to make it more particular to the problem at hand:

Element	Description
Software component	System hosted at EMA
Operations	Create, read, update, and delete
Inputs	Search terms, documents, metadata attributes
Outputs	Documents, metadata attributes
Underlying types	e.g. MedicinalProductDefinition, SubstanceDefinition, Task etc. (see section 7.)

4.3. *What the API is not*

It should also be noted that there are misconceptions and fallacies about an API, so an API is not:

- A software component that you install on a computer
- A process that automates human activities
- An end-to-end system between the NCAs and EMA

4.4. *Flexibility and constraints*

The definition of the API must be such that it addresses concerns of all the stakeholders as opposed to a small number of stakeholders. This is the trade-off between genericity and specificity, and to be able to specify an API, the following points must be taken into account:

- The API must meet the requirements.

[1] Based on Wikipedia: http://en.wikipedia.org/wiki/Application_programming_interface

- The stakeholders have different needs as they have different business processes, IT infrastructures and budgets.
- There will only be one API for all stakeholders.
- It is important to draw the line between generic features, usable by all stakeholders, vs. specific features, usable just by one or a few stakeholders only.
- Features that appear to be specific to one or a few stakeholders must be implemented on the client side and are out of the scope of the API definition.

4.5. Spelling

The resources and attributes in this specification as defined as per FHIR convention, which has standardised on US spellings. However, ISO IDMP has standardised on British spellings, which has been used within this specification in the descriptions and textual explanations. This generates a certain mismatch that is however unavoidable.

5. Specification

The specification for the API is based on the RESTful style API. The same style of API was adopted for the SPOR API 1.x, PSUR API and for the Common Repository; it will also be the style for the other components of SPOR. This is for the sake of consistency but also for its clarity, ease of use with minimal infrastructure and its clear separation between resources and the operations that can be applied on those resources.

5.1. *Versioning*

5.1.1. Relationship to SPOR API v1

This version 2 API is not a direct evolution of version 1, but is implemented with a new technology, FHIR. Version 1 interfaces will be maintained, and in some cases referenced from the V2 services.

V2 includes all Product and Substance related services, PMS, SMS, Change Requests for SMS and Translations. See 6.2. for a summary.

Both V1 and V2 use XML and JSON based RESTful APIs and are in many ways very similar. V2 uses standardised data structures and URL patterns from the FHIR specification but is very similar in principle and capable of easily co-existing with V1.

Especially, SPOR API 1.x resources and endpoints are expected to be used in combination with SPOR API 2.x:

- Document related SPOR API 1.x
- SearchQuery related SPOR API 1.x
- Any organisation and location identifiers are expected to originate in SPOR API 1.x for OMS.

Examples of how SPOR API 1.x and SPOR API 2.x can be combined are presented in section 9.1.

5.1.2. **SPOR API V2 versioning**

The SPOR API V2 is based on FHIR specification. Both SPOR API and FHIR will keep separate versioning scheme and evolve at own pace. Each version of the SPOR API V2 will be based on a particular version number of FHIR, as recorded in this document.

This API will be up-versioned (for example, to become V2.01) if and when changes are necessary, in a controlled and communicated manner. See http://hl7.org/fhir/directory.html for a list of version numbers of FHIR release. FHIR servers communicate their supported version as part of their conformance statement. See http://hl7.org/fhir/capabilitystatement.html.

5.1.3. **XML schemas versioning**

FHIR APIs are not versioned at schema level. A given FHIR server shows its version and properties using its CapabilityStatement resource (see http://hl7.org/fhir/capabilitystatement.html).

FHIR schemas are issued with a release of the FHIR standard, via the downloads page of the relevant release of FHIR (http://hl7.org/fhir/downloads.html). These schemas will not be altered, although future releases of the API may adopt newer iterations of the standard. Variation is accommodated by supporting different subsets of the elements in these the FHIR models and schemas, and the use of extensions (see below). FHIR allows for validation beyond what is possible with XML schema, including schematron and FHIR profile-based validation. SPOR API V2 will come with its own profiles to validate FHIR resources, and these profiles will be specific for the version of the SPOR API specification.

XML document instances should not include a schemaLocation attribute, since this can be file system dependent and not transportable between systems. Modern XML tools support schema validation without the use of schemaLocation in instances.

5.1.4. **Service versioning**

Each endpoint URL will be prefixed with /v{version}, where version is the service version number. The service URL is case sensitive.
i.e. `GET /v2/MedicinalProductDefinition`

Where a breaking change is required, a new versioned endpoint will be released. The previous version will be supported for a specific duration.

5.2. ***Authentication and authorisation***

All SPOR services require authentication, unless explicitly stated otherwise in the service definition (see *6. REST Services* for details).

Authentication will be HTTP Basic Authentication over SSL.

Authorisation will be RBAC with roles assigned during user registration.

5.3. ***FHIR extensions***

FHIR deliberately does not cover every localised detail of every healthcare domain. Specifically accommodating every last information point for the world's diverse healthcare data items would make the FHIR core unmanageably large and complex. Instead FHIR defines the most commonly used subset

of data items and lets individual implementation extend this, in a controlled, enforceable and well documented manner. For more details see http://hl7.org/fhir/extensibility.html. Some data items within this API use FHIR extensions, and these are documented within the individual specifications for those resources as used in this API.

5.3.1. Multi-lingual Strings

Some text from a product's data may appear in multiple languages. A standard FHIR extension, to the string datatype allows a set of translated strings to be attached. See http://hl7.org/fhir/extension-translation.html.

An example of it in use:

```
<?xml version="1.0" encoding="UTF-8"?>
<MedicinalProductDefinition xmlns="http://hl7.org/fhir">
    <elementExample value="c'est une chaîne">
        <extension url="http://hl7.org/fhir/StructureDefinition/translation">
            <extension url="lang">
                <valueCode value="en-gb"/>
            </extension>
            <extension url="content">
                <valueString value="this is a string"/>
            </extension>
        </extension>
        <!-- the whole extension can be repeated for more languages -->
    </elementExample>
```

Note that this is a "complex extension" - an extension that consists of two other extensions (because it is a pair, the language and the string itself).

The first string, in the example above "c'est une chaîne", is in the language of the resource as a whole, which can be specified in the inherited attribute Resource.language.

Implementation guidance will be given as to which specific elements can be extended in this way in the SPOR API.

5.4. *HTTP methods*

The API makes use of the standard HTTP methods such as `GET` and `POST` to read and write respectively from and to PMS and SMS servers.

These are described in detail as part of the standard FHIR specification: http://www.hl7.org/fhir/http.html, with a summary at http://www.hl7.org/fhir/http.html#summary

5.5. *HTTP errors and status*

The API will make use of a number of HTTP status codes where applicable.

See: http://www.hl7.org/fhir/http.html#2.21.0.4

and here: http://www.hl7.org/fhir/http.html#summary

Not all of the above referenced HTTP codes are used in this API.

Those that are used:

Read (GET), Update (PUT)	200	OK	
Create (POST)	201	Created	
Update (PUT)	202	Accepted	For an asynchronous operation, indicates initial basic success, with more work ongoing
Delete (DELETE)	204	Success and No Content	Success (no data needs to be returned in the body. Compare to 200, which usually returns the created data). Deleting a resource that doesn't exist gives a 204, not a 404.
Search (GET), Update (PUT), Create (POST)	400	Bad Request	Resource (PUT, POST) failed basic validation or search parameters (GET) failed basic validation, or no id provided (PUT)
All	401	Not Authorized	Operation needs authorization and no authorization was attempted
All	403	Forbidden	Operation needs authorization and authorization failed
Read (GET), Search (GET), Update (PUT), Create (POST)	404	Not Found	Unknown resource (for GET, POST), or unknown resource type (for Search, PUT)
Update (PUT), Delete (DELETE)	405	Method Not Allowed	Can't update a resource that didn't exist (PUT. Note this is different to 404, since in some other systems PUT can be allowed as a way to create). Or not permitted to delete (DELETE).
Update (PUT), Create (POST)	422	Unprocessable Entity	the proposed resource (while basically valid) violated applicable FHIR profiles or server business rules

Note that Updates will never create a record that didn't exist before.

Some POST operations are used with transaction Bundles. These insert or update multiple resources, and return a Bundle of transaction results, each having an HTTP result code (see 5.7.1.)

Whenever there is any sort of failure, in addition to an appropriate HTTP response code, extra information will be returned in an FHIR OperationOutcome resource. See http://hl7.org/fhir/operationoutcome.html

5.5.1. **Asynchronous Updates**

All update operations may be completed asynchronously. These may initially return 202 Accepted (rather than 200 OK), for a success code, and provide a status URL in the Content-Location header. This URL can be polled to check if the operation has completed. Assuming a successful eventual outcome, at some future time a GET to the status URL will result in a 20X success code (200 or 201), combined with any results as would have happened in the synchronous case (e.g. the ids of created resources).

The pattern is documented here http://hl7.org/fhir/R4/async.html.

5.5.2. **Warnings**

As a service to the client, when requested by the caller, some operations will succeed but will return an OperationOutcome even in the event of success, that gives some warnings or hints about the validation state of the resource, completeness etc. These are documented alongside each service. To activate this response the caller must use the HTTP prefer header:

```
Prefer: return=OperationOutcome
```

5.6. ***FHIR References and Identifiers***

FHIR uses two separate types of identifiers, known respectively as the "id" and the "identifier". These sound similar but are significantly different and it is important to distinguish their purpose and use. The id, of which there is only one, is how the resource is accessed on a technical level (record read, write, location), and is specific to the FHIR interface. The identifier(s) are human readable strings that used as working numbers for the day to day identification of products and substances and exist outside of FHIR. Both ids and identifiers are strings and can be numeric or alphanumeric if desired.

The two types are needed because the uses are very different. Ids are only for internal software use of the API. Identifiers are for human users of the software system. It is possible in theory for the id to be the same string as the identifier. This is an attractive idea but has problems in practice. Every resource has to have one persistent id that never changes, from the moment the resource is first created. Some products have an MPID identifier that could also be used as an id, but others do not (e.g. draft resources, investigational products). The MPID has dependencies on the properties of the product, which in some scenarios may change, even within a single resource that records that product. In general therefore, the id will not be numerically equal to any identifier.

5.6.1. **FHIR resource id (1..1)**

This is the RESTful id of a resource, which corresponds to its location on the server, and can be used to directly access the resource. This is not a business identifier. It is a technical identifier and should never be exposed to a standard user. The digits/letters usually have no "real world" meaning, and don't correspond to any another number. There can only ever be one id per resource, and in normal operation it never changes. It is only ever used in the context of the FHIR interface and is always generated by the server, never the client or a user.

FHIR data consists of a set of resources, normally on a "RESTful", web-based server. Each resource can be thought of as a web page, and it has an id that can be considered as its location.

e.g. /server/MedicinalProductDefinition/4be6d0b5-9d39-4367-9c6d-ed030790db01

where 4be6d0b5-9d39-4367-9c6d-ed030790db01 is the FHIR restful id. (The id is shown here as a UUID, but it need not be - other formats are equally possible.)

A software system that accesses the URL above will directly see the data for that single resource (or an error if the id is not recognised). This is a direct read access, with no searching or retrieval of other connected resources. Note that standard users will not normally see or interact with these ids and will not ever see the URLs that software uses internally. All URLs and ids will normally be hidden within the GUI software that provides the working screens to the business user. Hence the length of these ids will not be an issue in day to day use.

The id can be considered as metadata, because it is not part of the product (or substance etc.) data itself – it's just a record id, or a database id. In FHIR terms the id is not defined on a per-resource basis but is inherited from the base class of all resources: Resource. It is therefore documented separately from each resource (and can be easy to overlook). For example it is not shown in the list of elements here: http://hl7.org/fhir/documentreference.html#tabs-struc, but instead is covered here: http://hl7.org/fhir/resource.html#tabs-struc.

5.6.2. **FHIR resource identifier (1..*)**

This is the business or "real world" identifier of the resource. Unlike the id, there can be multiple different identifiers per resource, and each may correspond to some string that is well known and used in other business processes. An example is an IDMP MPID. A product, or substance, may be known by different numbers in different catalogues, or jurisdictions. These can all be stored as identifiers.

The Medicinal Product identifier is shown here:
http://build.fhir.org/medicinalproductdefinition.html#tabs-struc

Identifiers are not the primary way that software accesses a resource in a RESTful system. This is partly because they do not have to be unique to one resource. (It is possible that an approved product and an un-approved draft of a new version of it would share the same identifier, but it has to have a different id.) Identifiers can be used in queries.

A query that find resources with a given identifier:

```
/server/MedicinalProductDefinition?identifier=1000041569
```

Note that this is a query, not a direct read (see above), so it uses the "?{field}={value}" syntax (see http://hl7.org/fhir/search.html, and also the specific search parameters defined for each resource in this specification). This also means that it may return more than one resource, and therefore the results are always contained in a FHIR resource Bundle (see 5.7.).

In particular, note that it is *not* possible to do this:

```
/server/MedicinalProductDefinition/1000041569
```

because the identifier cannot be used as the RESTful id.

Although identifiers are not the primary index for data in a FHIR system, they will typically be the method that users enter to access data. The software user interface issues all the URL queries for them, without them needing to know any technicalities.

Unlike ids, identifiers always have a "system" that says what sort of identifier they are (e.g. MPID, external product id, PCID etc). This is part of the "Identifier" data structure (note upper case "I") that every identifier uses (see http://hl7.org/fhir/datatypes.html#Identifier).

If an identifier string may not be unique across all types of identifier, the specific type can optionally be referenced e.g.

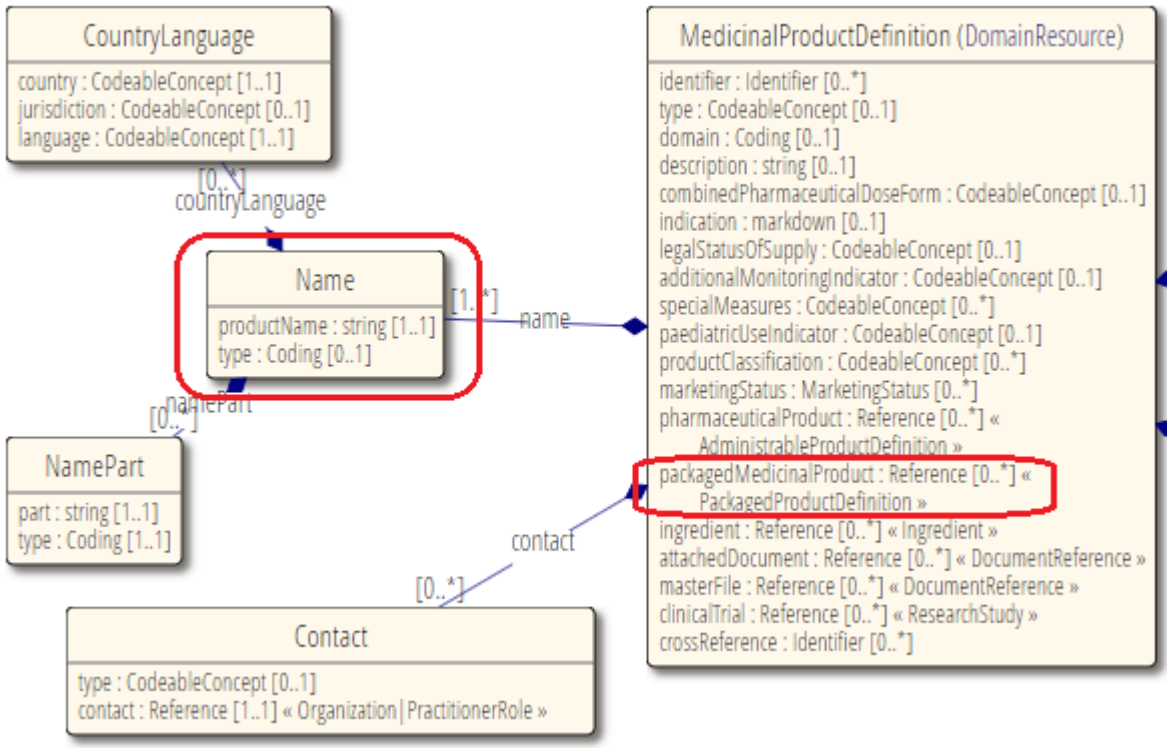
```
/server/MedicinalProductDefinition?identifier=http://ema.europa.eu/fhir/MPID|1000041569
```

5.6.3. **References**

FHIR resources can be thought of as pages, and these pages have "references" between them that act like hyperlinks. See http://www.hl7.org/fhir/references.html.

It is usually necessary to use more than one FHIR resource type to represent some useful collection of data items. This involves having several resource types, and using the RESTful id of one as a reference

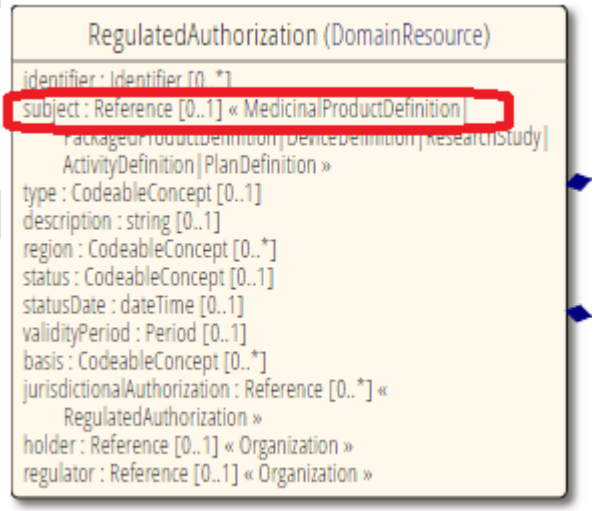
in another. In the following example, Name is directly a part of the MedicinalProductDefinition resource, but the Package is not (it is a reference):



In business terms one resource may be considered the parent and another the child. It is natural to think of the Medicinal Product as being the "parent" of the packages or the indications, etc.

In technical terms things can be linked in either direction, depending on what is most convenient, and reduces the amount of updates.

This second resource “points” to the first, using a reference, and via its id (which is really a page location), as in the following example:



This resource-oriented, or page-oriented view of data has implications for Medicinal Products particularly, because the full product data set is split over a series of resources, connected by

references. Subject to business constraints of what makes sense and is allowed, these can be treated either as separate resources, retrieved and updated individually, or as a group of resources accessed together. This is discussed in the overview of product services in section 6.4. One mechanism for allowing a group of resources to be kept together is the FHIR Bundle.

5.7. ***Bundles***

The API makes use of the FHIR Bundle resource. See http://www.hl7.org/fhir/bundle.html.

A Bundle is a container resource, that is used whenever a group of more than one resource is needed.

Each bundle has some basic header information, including its type (searchset, transaction, transaction-response, batch), and a total number of "hits", for a search. It then consists only of a repeating "entry" structure, which contains one resource, of any type, and possibly a request or result section, for use with transactions.

Some examples are:

- Search results

  Receive a bundle of 0 or more resources, of the type requested, and possibly other types that are linked to that type (requested using "_include"). Also used when multiple resources are retrieved using an operation such as $everything

- Transaction

  Used when a linked set of resources must be created or updated. It acts like a repeated RESTful call, all in one call. This allows for "atomicity" and referential integrity (if any parts fail, every part is rolled back). Also provides a way to link different resources correctly when creating a set that must reference each other by ids. These ids are normally server assigned, and the client doesn't know them in advance. Bundles that have items linked via temporary ids get these automatically replaced with the real ids when the data is saved.

- Transaction-response

  A transaction is a way of performing several http calls at once. In this case several sets of http results are needed together, and this uses a transaction-response bundle.

- Batch

  In some cases a service may accept a set of resources to be processed, but with no requirement for transactional behaviour or link resolving.

Bundle general schematic:

```
Bundle
    type= transaction|transaction-response|searchset|batch
    total=N (for searchset)
    [entry
        {resource}
        [request (used in a transaction to give http commands)
            method=POST|PUT|GET|DELETE
        ]
        [response (used in transaction response to give http results)
            location={URL of resource, including id}
        ]
    ] *
```

5.7.1. **Transaction Bundles**

Detail schematic of a transaction Bundle, showing how linkages work:

```
Bundle
    type=transaction
    entry
        {Parent Resource Type}
            {attribute of child resource type}
                reference=temporaryUuid (temporary local id of child, gets replaced by
server)
        request
            method=POST
    entry
        fullUrl=temporaryUuid (matching temporary local id)
        {Child Resource Type}
        request
            method=POST
```

The result would be in this form:

```
Bundle
    type=transaction-response
    entry (one per created resource, same order as incoming transaction)
        response
            location={ParentResourceType/parentid} (URL says type and id of created
resource}
    entry
        response
            location={ChildResourceType/childid}
```

The created parent resource will have a reference in it that points to {`ChildResourceType/childid`}

5.7.2. **Bundle endpoints**

Bundles can be of a mixed set of resource types. For this reason, Bundles being sent to the server are posted to the root of the server (e.g. /v{version}), rather than to a specific resource type endpoint.

Bundles can also be retrieved from the other, resource specific, endpoints however. For example, a search on `/v2/MedicinalProductDefinition` would return a set of MedicinalProductDefinitions in a searchset Bundle.

5.8. ***Searching***

Search capabilities are offered on every resource based on GET operations, using a number of query parameters.

e.g. `GET /v2/RegulatedAuthorization?status=pending`

Each resource search endpoint will list a number of recognized query parameters that can be used to filter the results of a search. Out of all the possible query parameters, a maximum of 10 can be provided in a single search, in no particular order.

It is also worth noting that searches will be performed against the latest version of the resource that the caller is authorised to view as per 9.2.1. . No match against historical information will be considered, but history can be accessed via the "version" operations (e.g. Get Product Version").

5.9. *Paging and sorting*

All FHIR results from a server are subject to paging. This is described here:
http://hl7.org/fhir/http.html#paging

This only affects results where there is more than one result (i.e. searches). It is possible to override the default page size, by asking the server to supply more records per page using the "_count=N" search parameter modifier.
e.g. `GET /v2/RegulatedAuthorization?status=pending&_count=100`

As a special case "_count=0" can be used to indicate "all resources".

As documented in FHIR, paging works by each "page" of search results (a Bundle), having links to the first, last, next and previous pages. Implementations only need to use these supplied links, from the Bundle header, to navigate the entire search results. In technical terms, this operates by the caller using the appropriate URL, which contains a search token that is unique this search result set, and a page number.

e.g. `GET /v2/RegulatedAuthorization?searchtoken=abc123&page=3`

Knowing this allows a client to construct the URL for any page in the search results. However there is no requirement to be able to parse and use the token, because the necessary URLs provided can be used to reach each page in turn.

Searches can be sorted using the "_sort" parameter, as described here:
http://hl7.org/fhir/search.html#_sort

5.10. *Resources and representations*

For an API with a RESTful style, a resource is anything that can be identified and manipulated by a set of HTTP verbs. Resources are defined by FHIR and referenced in the services in the rest of this document, in particularly in section 7.

Not all of those resource types will be directly exposed as RESTful endpoints – some are only used embedded within others. Resources can be expressed using various representations depending on the need of the user and the nature of the resource. In the context of this API, the representations for resources are, according to their media type defined by IANA:

- **`application/fhir+xml`** - used to indicate that the resource is represented by xml data.
- **`application/fhir+json`** - used to indicate that data is represented using the JavaScript Object Notation, which is a programming language independent data format, expressing information in the form of key-value pairs.

The default resource representation is application/fhir+xml and it is the client's responsibility to indicate if application/fhir+json is required. For this purpose, the client must make use of the `Accept` header field in the HTTP request.

If the representation requested is not supported by the server then an appropriate error is returned by the server to the client (see section 5.5.).

Examples:

- Request for a resource representation in xml format: (may be omitted as default)
  `Accept: application/fhir+xml`

- Request for a resource representation in JSON format:
  `Accept: application/fhir+json`

See *6. REST Services* for the Accept Headers supported by each service.

5.10.1. Encoding

All SPOR resources are UTF-8 encoded, unless explicitly stated otherwise in the service definition (see *6. REST Services* for details).

5.11. *Request parameters and searches*

For this API specification, the parameters for a request can be provided in number of ways to the server:

- **Path**: `/v2/[type]/{id}`
  where the single parameter is the resource's FHIR id. [type] represents the name of a type of resource e.g. MedicinalProductDefinition. Note that resource names in FHIR are always case sensitive and in upper camel case.
- **Query string**:
  `/v2/[type]?{param}={[op]value[,value]}[&{param}={value}]`

  e.g.: `/v2/[type]?name=example,exampletwo&_count=100`
  where the resource type is followed by a name-based query and a request for up to 100 records per page

  `[op]` represents possible use of other operators than "=". See http://hl7.org/fhir/search.html#prefix

  `[,value]` represents possible use of comma separated values for "or"ed criteria.

  `[&{param}={value}]` represents use of multiple query phrases, which are logically "and"ed together, or the use of extra query modifiers such as _count, _format, _sort.

  The actual parameters that can be used are defined for each part of the API, see for example the query parameters in Search Product 6.4.2.

  See also http://www.hl7.org/fhir/http.html#search
  and http://www.hl7.org/fhir/search.html
- **Header of the request**: for example: `Accept: application/fhir+json` which is used by the server to determine which representation will be return to the client (in this case overriding the default of XML).

All of the above can be used jointly in the same request to the server. The service URL is case-sensitive.

5.11.1. Parameter characteristics

In the definitions below all Endpoint path parameters are mandatory, unless shown in square brackets ([]).

String based searches in FHIR are by default case and accent insensitive, and a field matches a search string if the value of the field equals or starts with the supplied parameter value. In other words, "starts with" is assumed.

The :contains modifier can always be added to allow full substring searching. :exact can be used to restrict to exact matches in terms of string position and case sensitivity.

For full details of how query parameters work in FHIR see http://www.hl7.org/fhir/search.html.

To prevent excess server load, for this API the number of search parameters per URL is limited to 10.

5.11.2. **Full text search**

The FHIR API supports searching on the text of multiple fields using the "_content" parameter. See http://www.hl7.org/fhir/search.html#content. The specific subset of fields is yet to be provided.

5.11.3. **Chained searches**

Medicinal Product records are stored in FHIR as multiple resources, linked by FHIR references. For example, a MedicinalProductDefinition resource will have references to its packs, in a set of referenced PackagedProductDefinition resources. The API exposes endpoints for all resources, including MedicinalProductDefinition and its "parts" – PackagedProductDefinition, RegulatedAuthorization etc.

Each endpoint can be queried. For products this is documented for the main "parent" resource MedicinalProductDefinition in Search Product 6.4.2. and as a group for all the "child resource" parts in Search Product Part 6.4.3.

It should be noted that it is possible in FHIR to use search parameters from child resources even when querying on the parent resource. This is known as a chained search and is described here: http://www.hl7.org/fhir/search.html#chaining

An example in schematic form would be:

```
GET /v{version}/MedicinalProductDefinition?{parent-param}={value}&{child-attribute-name}.{child-param}={value}
```

In the above child-attribute-name is the name of the child resource when used as an attribute in the parent resource. For example, the MedicinalProductDefinition.masterFile is a reference to a DocumentReference resource.

A chained query could be:

```
GET
/v{version}/MedicinalProductDefinition?domain=human&packagedMedicinalProductDefinition.marketingStatus=pending
```

Note that although chained queries can ask questions about data in a linked child resource, this is still a query exclusively on the MedicinalProductDefinition resource and so will only return MedicinalProductDefinition resources and not PackagedProductDefinition resources – but also see "_include" below.

5.11.4. **Including other resources in search results**

Every FHIR resources endpoint can be queried, as described elsewhere, and using the specific parameters defined in this API. But each resource endpoint normally only fetches query results for that particular resource type, not any others that may be linked to that data. Since the data for a single product data is spread across a series of resources, for convenience the product API defines the $everything operation (see 6.4.4.). Generally, for more information about FHIR operations see
http://www.hl7.org/fhir/R4/operationslist.html

However, when searching (e.g. via Search Product, 6.4.2.) it is possible for the results to include extra resource "parts" that are linked to the parent. This uses the "_include" or "_revinclude" parameters. See
http://www.hl7.org/fhir/search.html#include

An example would be:

```
GET
/v{version}/MedicinalProductDefinition?domain=human&_include=MedicinalProductDefinition
:masterFile
```

Note that this includes the child resource by using the attribute name of it as used by the parent. It does not use the name of the child resource type (which would be DocumentReference). The result will be a Bundle containing matching resources of type MedicinalProductDefinition and their child DocumentReference resources, covering the attached master files.

Reverse include (_revinclude) is that same as _include, but is for when the "child" resource is not actually referenced from the parent, but instead there is a "reverse" reference from the child to the parent. Typically this uses a link in the form child.subject=parent-reference (rather than parent.child-attribute=child-reference) and is used in most of the links between parts of the product.

An example would be:

```
GET
/v{version}/MedicinalProductDefinition?domain=human&_revinclude=RegulatedAuthorization:
subject
```

This will return the matching human MedicinalProductDefinitions and any RegulatedAuthorization resources that have those products as their subject.

With $everything, chaining and _include/_revinclude it is possible to do some very flexible joins between resource types.

5.12. ***Metadata***

The metadata associated with resources is documented here:
http://hl7.org/fhir/resource.html#metadata.

Metadata includes the resource "versionId" and "lastUpdated" date. Both are available on every resource.

"versionId" is the number of the FHIR history version. This applies to all resources, but for details see Product Versioning 6.4.1. This is incremented with each save of a resource. It cannot be queried directly, but is instead accessed by the "_history/{version-number}" method.

"lastUpdated" is the server date of the last change to any data item in the resource and is also therefore the date of the last "save". It can be queried using a special query parameter called "_lastUpdated". This works the same as any other query parameter and is documented with examples here: http://www.hl7.org/fhir/search.html#all. See also example in Search Substance 6.5.2.

These items are defined for every FHIR resource, because they are in the Resource class, which is the base type of all resources. They appear in the full XML or JSON representation, when the resource is returned from a server, although, being server assigned, they are usually omitted when sending data to a server.

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<MedicinalProductDefinition xmlns="http://hl7.org/fhir">
    <id value="4be6d0b5-9d39-4367-9c6d-ed030790db01"/>
```

```
<!-- metadata is near top of resource -->
<meta>
   <versionId value="2"/>
   <lastUpdated value="2018-10-27T18:40:21Z"/>
</meta>
<!-- metadata is followed by the resource proper -->
<identifier>
   <system value="http://ema.europa.eu/example/MPID"/>
   etc.
```

5.13. *Standards*

- All dates/times returned or provided as path parameters must be expressed in the timezone UTC and comply with the formatting of the allowed formats of the ISO-8601 standard.
- The API supports a maximum URL size of 2048 characters – including the hostname, resource path and query parameters. This limit is subject to ongoing technical investigations.

6. REST Services

6.1. *Resource Summary*

A full list of resources for this API is (for full description please see 7. Resources):

Resource	Description	SPOR Domain
MedicinalProductDefinition	Detailed definition of a medicinal product, typically for uses other than direct patient care (e.g. regulatory use).	**Product**
RegulatedAuthorization	RegulatedAuthorization is a resource covering the Marketing Authorization of a Medicinal Product, from a regulatory point of view.	**Product**
ClinicalUseIssue	Facts about a particular medication in relation to its intended use (indication), situations where it should not normally be used (contrainindication), known side effects (undesirable effects), and clashes with other substances - medications, foods etc (interactions)	**Product**
Ingredient	An ingredient of a manufactured item or pharmaceutical product.	**Product**
PackagedProductDefinition	A Medicinal Product in a container being part of a package, representing the entirety that has been packaged for sale or supply	**Product**
AdministrableProductDefinition	A pharmaceutical product described in terms of its composition and dose form.	**Product**
ManufacturedItemDefinition	A manufactured item, as contained in a packaged medicinal product.	**Product**

Resource	Description	SPOR Domain
DeviceDefinition	The DeviceDefinition resource is used to describe the characteristics and capabilities of a medical device.	**Product**
SubstanceDefinition	The detailed description of a substance, typically at a level beyond what is used for prescribing.	**Substance**
Task	A task resource describes an activity that can be performed and tracks the state of completion of that activity.	**Substance**
DocumentReference	A DocumentReference resource is used to describe a document that is made available to a healthcare system.	**Substance, Product**
Bundle	A container for a collection of resources.	**Substance, Product**
OperationOutcome	A special resource that is used only to show error, warning or information messages, as a result of an attempted action.	**Substance, Product**
CapabilityStatement	A Capability Statement documents a set of capabilities (behaviours) of a FHIR Server.	**Substance, Product**

6.2. *Service Summary*

Service	Description	Details	SPOR Domain
Product Service	This service enables the user to create, update and view products (MedicinalProductDefinition and other resources) and allows the user to search for products on the provided search criteria.	*Product Service*	**Product**
Change Request SMS Service	This service enables the user to create and update change requests to request the creation/update of a substance (via a SubstanceDefinition resource). Services are not provided to create/update/delete SubstanceDefinitions directly. All such operations are facilitated via change requests services. Each change request must include a Request Reason. This informs the Data Steward the type of change being requested. The justification attribute can be used to provide more information about the reason for the change request, and to	*Change Request SMS Service*	**Substance**

Service	Description	Details	SPOR Domain
	provide any supplementary information. This service also enables the user to retrieve a single change request via one of its identifiers, or a collection of all the change requests that the user has created. A user is only able to retrieve their own change requests; they are not able to retrieve change requests raised by other users.		
Substance Service	This service enables the user to retrieve a single substance (SubstanceDefinition) via one of its identifiers, or a collection of substances based on provided search criteria. Direct creation or updating of substances is not allowed (See the Change Request SMS Service). Language translation of Substances is also possible.	7. Substance Service	**Substance**
Substance Translation Service	This service allows an excel sheet of translations of substance names to be downloaded or uploaded.	*Substance Translation Service*	**Substance**

6.3. *Validation Operation*

In adding to the services below, each resource type will also support a validation endpoint (excluding ones that are not normally exchanged, such as CapabilityStatement, but including Bundle). This endpoint will accept a resource and check its validity, returning an OperationOutcome that says if the resource is considered valid by this server, and what issue there is if not. It does not ever actually save the data and can be considered as a test mode. This service returns a 200 OK if the validation is successfully *attempted* (but this doesn't mean it is a successful validation).

See http://hl7.org/fhir/resource-operations.html#validate

Resource Information

Endpoint	POST /v{version}/{resource-type}/$validate
Request	
Accept	application/fhir+xml application/fhir+json
Body	<resource> (matching the endpoint used)
Content-Type	application/fhir+xml application/fhir+json
Response	
Body	OperationOutcome

Path Parameters

Name	Description
version	Service version number **Example value: 2**

Query Parameters

None

Example Request

```
POST /v2/MedicinalProductDefinition/$validate
```

6.4. *Product Service*

This service enables the user to create and view products (MedicinalProductDefinition and other resources) and allows the user to search for products on the provided search criteria.

The full data for a medicinal product is split across several FHIR resources, with linkages between them (FHIR references). FHIR endpoints and services are normally single resource specific.

Exceptions to this are:

- when a transaction bundle is used to update several linked resources at once
- when using "_include" or "_revinclude" parameters with searches, to fetch the main resource and the "included" linked ones
- when using the $everything operation to fetch the main resource and all the linked ones.

This all means that the product data is accessible in several ways, as a whole, or as a series of parts. The "main" resource is MedicinalProductDefinition. The linked resources, or parts, are these:

RegulatedAuthorization
ClinicalUseIssue
Ingredient
AdministrableProductDefinition
PackagedProductDefinition
ManufacturedItemDefinition
DeviceDefinition

The services are named "Part", if they operate on these parts individually. Each "Part" service will work with any of these types, and the list is not repeated each time but referred to as `"{Product Part Resource Type}"`, to act as a placeholder.

The "part" services, e.g. Search Product Part 6.4.3. , Update Product Part 6.4.15. , are documented like this:

Get Product Part:

```
GET /v{version}/{Product Part Resource Type}/{resource-id}
```

There is some variation between the different services, but the key thing here is "`{Product Part Resource Type}`". This means any of the "part" resources can be used and the name of the appropriate resource is to be used in the URL. To access product parts for authorisations, the RegulatedAuthorization resource is used. So the URL to use above becomes, as an example:

```
GET /v2/RegulatedAuthorization/fa5a7413-a19e-4524-8fd6-ced86f64038b
```

This gets a resource of type RegulatedAuthorization, and with REST id fa5a7413-a19e-4524-8fd6-ced86f64038b. It returns a single resource, with no linked resources, or any Bundle wrapper.

"`/v2/RegulatedAuthorization`" is a FHIR endpoint, for the resource type RegulatedAuthorization, and it will support all the operations that are defined in this API – get, search, create, update, get version etc.

There is also an endpoint /v2/ClinicalUseIssue, and /v2/PackagedProductDefinition and so on, but each service only documents these in a non-specific way, which applies no matter which resource type is used with it.

6.4.1. Product Versioning

Products have complex lifecycles. These issues are addressed here. Unless stated otherwise all Product API features work on the most current version of a Product only.

6.4.1.1. *Updates Pending Approval*

The owner of a product can make updates to it, and save them to the server, for regulatory approval. During the time between the update and approval being granted, these updates will be hidden from other users. By design, any read of the latest version of resource may return different data depending on user permissions. This has no technical impact on the API definition.

6.4.1.2. *Older Versions of Products*

Previous versions of product are maintained. These can be accessed via the standard FHIR history mechanism (see http://hl7.org/fhir/http.html#history). Note however that there is only one "current" version of each resource. Old versions exist in the history, but these will not be found by searches, and are only seen when explicitly accessed using "_history". Draft versions of product are a separate copy of the resource and exist alongside the current version.

See also Get Product Version 6.4.7. and Get Product Versions 6.4.8.

6.4.1.3. *Draft Versions of Products*

A draft in this context is a temporary "working" copy of a product, being edited, with the intention that the edits will be merged back into the main product at some later time. In API terms these are a new resource instance, created from the product's data, but separately managed. They are not "old" versions of products and are not accessed via "_history".

A draft copy can be created from a product using the $createdraft operation. This makes a new copy on the server and returns an id to the caller. After that, the draft product is accessed the same way as any other product resource. There are no special operations, and it just uses its id, in the normal way. These ids will be distinct from that of the original product (unlike for older versions see 6.4.1.2. Draft

instances will share the same product business identifier (which can be used to cross reference them) and will have another identifier to give a draft version number.

6.4.2. **(EP301) Search Product**

Use this operation to return a collection of products, based on provided search criteria.

This also supports returning the linked "part" resources of the product, by selectively using the "_include" or "_revinclude" parameter.

This operation supports server-side paging (See Sections 5.8. & 5.9.)

Resource Information

Endpoint	GET /v{version}/MedicinalProductDefinition?{param}={value}[&{param}={value}]
Request	
Accept	application/fhir+xml application/fhir+json
Body	n/a
Content-Type	n/a
Response	
Body	Bundle of MedicinalProductDefinition(s), and optionally other types, if parameter "_include" is used e.g. Bundle Total=N [entry MedicinalProductDefinition] * [entry {Product Part Resource Type} (if present and specified by "_included")] *

Path Parameters

Name	Description
version	Service version number **Example value: 2**

Query Parameters

Name	Description
name	The product name
identifier	A product identifier such as MPID or EV Code
name-language	Medicinal Product Name Language
domain	Human / Vet
type	Approved or Investigational product
product-classification	Classification of the product as per various system, such as for example ATC (but not only)

Name	Description
contact	Contact name, for example, a QPPV's name
master-file	Identifier of the master file or organisation that holds it
_has:RegulatedAuthorization:subject:identifier	Marketing Authorisation Number and other primary identifiers
_has:RegulatedAuthorization:subject:status	Authorisation status
_has:RegulatedAuthorization:subject:region	Country or group of countries where the authorisation has been granted
_has:RegulatedAuthorization:subject:holder	Marketing Authorisation Holder (and other authorisation holders)
_has:RegulatedAuthorization:subject:case	Procedure or application number
_has:RegulatedAuthorization:subject:case-type	Procedure or application type
_has:PackagedProductDefinition:subject:identifier	PCID : Medicinal Product Pack Identifier
_has:AdministrableProductDefinition:subject:dose-form	The administrable dose form, after necessary reconstitution
_has:AdministrableProductDefinition:subject:route	The path by which the pharmaceutical product is taken into or makes contact with the body
_has:AdministrableProductDefinition:subject:target-species	Target species that the pharmaceutical product is to be administered to
_has:PackagedProductDefinition:subject:manufactured-item:dose-form	Dose form as manufactured and before any transformation into the pharmaceutical product
_has:ClinicalUseIssue:subject:indication	The situation that is being documented as an indication for this item
_has:PackagedProductDefinition:subject:device:identifier	Device unique identifier
_has:PackagedProductDefinition:subject:device:type	Kind of device or device system
_include	The name of an attribute that links to another resource that is a part of this products set of resources e.g. "MedicinalProductDefinition:masterFile". Note that this is the name of an attribute of MedicinalProductDefinition and not the name of the resource type (which would be DocumentReference) Usage: _include=MedicinalProductDefinition:masterFile This causes the linked DocumentReference to also be returned in the results bundle.
_lastUpdated	Searches on the modified date of the data e.g. _lastUpdated=ge2018-01-01&_lastUpdated=le2018-12-31

Example request

```
GET /v2/MedicinalProductDefinition?name=name&status=NON_CURRENT&_sort=name
GET /v2/MedicinalProductDefinition?identifier=http://ema.europa.eu/fhir/EVCode|X
GET /v2/MedicinalProductDefinition?identifier=http://ema.europa.eu/fhir/MPID|X
GET /v2/MedicinalProductDefinition?_has:RegulatedAuthorization:subject:identifier=X
GET /v2/MedicinalProductDefinition?_has:PackagedProductDefinition:subject:identifier=X
GET /v2/MedicinalProductDefinition?_has:RegulatedAuthorization:subject:status=X
GET /v2/MedicinalProductDefinition?_has:RegulatedAuthorization:subject:country=X
GET /v2/MedicinalProductDefinition?_has:RegulatedAuthorization:subject:holder:name=X
```

6.4.3. **(EP302) Search Product Part**

Use this operation to search for a specific part of the data of a medicinal product. This allows individual access to the linked components that go to make a whole medicinal product.

The id used will typically be obtained from a reference within a MedicinalProductDefinition resource, or whichever resource is referencing the requested item.

This operation supports server-side paging (See Sections 5.8. & 5.9.)

Endpoint	GET /v{version}/{Product Part Resource Type}?{param}={value}[&{param}={value}]
Request	
Accept	application/fhir+xml application/fhir+json
Body	n/a
Content-Type	n/a
Response	
Body	Bundle of <Product Part Resource Type>(s) e.g. Bundle Total value=N [entry {Product Part Resource Type}] *

Path Parameters

Name	Description
Version	Service version number **Example value: 2**

Query Parameters

These are specific to each type

Resource	Name	Description
RegulatedAuthorization	identifier	Authorisation Number (for example, a Marketing Authorisation Number but also an Homeopathic Registration Number, etc.)
RegulatedAuthorization	status	Authorisation status
RegulatedAuthorization	region	Country or group of countries where the authorisation has been granted
RegulatedAuthorization	holder	Marketing Authorisation Holder (and other authorisation holders)
RegulatedAuthorization	subject	MedicinalProductDefinition that this resource belongs to
RegulatedAuthorization	case.identifier	Procedure number
RegulatedAuthorization	case.type	Procedure type
RegulatedAuthorization	case.application.identifier	Application number
RegulatedAuthorization	case.application.type	Application type
PackagedProductDefinition	identifier	PCID
Ingredient	substance	Code of the substance
Ingredient	specified-substance	Code of the specified substance

Resource	Name	Description
AdministrableProductDefinition	subject	The MedicinalProductDefinition that this resource belongs to
AdministrableProductDefinition	dose-form	The administrable dose form, after necessary reconstitution
AdministrableProductDefinition	route	The path by which the pharmaceutical product is taken into or makes contact with the body
AdministrableProductDefinition	target-species	Target species that the pharmaceutical product is to be administered to
PackagedProductDefinition	subject	MedicinalProductDefinition that this resource belongs to
ManufacturedItemDefinition	dose-form	Dose form as manufactured and before any transformation into the pharmaceutical product
ClinicalUseIssue	Type	The major type of this issue e.g. indication, contraindication, interaction, undesirable-effect, other
ClinicalUseIssue	indication	The situation that is being documented as an indication for this item
ClinicalUseIssue	contraindication	The situation that is being documented as a contraindication for this item
ClinicalUseIssue	effect	The situation in which the documented undesirable effect may manifest
ClinicalUseIssue	interaction	The type of the interaction e.g. drug-drug interaction, drug-food interaction, drug-lab test interaction
ClinicalUseIssue	subject	MedicinalProductDefinition that this resource belongs to
DeviceDefinition	identifier	Device unique identifier
DeviceDefinition	type	Kind of device or device system

Example Request

```
GET /v2/RegulatedAuthorization?status=pending
```

A common pattern is to find the parts relating to a particular product:

```
GET /v2/RegulatedAuthorization?subject={product-id}
```

6.4.4. **(EP303) Get Product**

Use this operation to return information for the core part of the data of a medicinal product, identified by its product-id. This is not the entire data set of the product, but a "header" resource, that contains links (FHIR references) to the other resources necessary to fully describe the product. See also Get Product Full 6.4.5. and Get Product Part 6.4.6.

Endpoint	GET /v{version}/MedicinalProductDefinition/{product-id}
Request	
Accept	application/fhir+xml application/fhir+json
Body	n/a
Content-Type	n/a

Response	
Body	Resource of type MedicinalProductDefinition

Path Parameters

Name	Description
product-id	A unique product identifier UUID **Example value:** 4be6d0b5-9d39-4367-9c6d-ed030790db01
version	Service version number **Example value: 2**

Query Parameters

None

Example Request

```
GET /v2/MedicinalProductDefinition/4be6d0b5-9d39-4367-9c6d-ed030790db01
```

6.4.5. (EP304) Get Product Full

Use this operation to return complete information for a specific product, identified by its product-id. Note that this operation is not subject to server-side paging. This operation is notable for bringing back a graph of connected resources, covering all the current information about a product.

Resource Information

Endpoint	GET /v{version}/MedicinalProductDefinition/{product-id}/$everything
Request	
Accept	application/fhir+xml application/fhir+json
Body	n/a
Content-Type	n/a
Response	
Body	<Bundle of MedicinalProductDefinition and other types> e.g. Bundle entry MedicinalProductDefinition [entry {Product Part Resource Type}] *

Path Parameters

Name	Description
product-id	A unique product identifier UUID

Name	Description
	Example value: 4be6d0b5-9d39-4367-9c6d-ed030790db01
version	Service version number **Example value:** 2

Query Parameters

None

Example Request

```
GET /v2/MedicinalProductDefinition/4be6d0b5-9d39-4367-9c6d-ed030790db01/$everything
```

6.4.5.1. *Use of $everything to clone a product*

It is sometimes necessary to create a new product record based on another one. The full product record is a connected graph of several instances of several resource types, so there are some considerations when doing this. Creating a product involves using temporary ids to link the resource parts, so that the server can know what is connected, before it assigns the permanent ids that it uses. For details see Create Product 6.4.12. and the references there.

To make a copy of a product, do the following:

Use $everything to fetch the latest version of the starting product (a bundle of resources).

Each resource will have a server assigned id, and references linking to other resources.

e.g.

```
<Bundle xmlns="http://hl7.org/fhir">
... snip ...
  <MedicinalProductDefinition>
      <id value="{product-id}"/>
... snip ...
  <RegulatedAuthorization>
      <id value="{authorisation-id}"/>
      <subject>
          <reference value="MedicinalProductDefinition/{product-id}"/>
      </subject>
```

Make any changes necessary for business reasons (edit the product).

When it comes time to save as a new product, use Create Product, and re-assemble a bundle to be POSTed. Add a fullUrl element to each resource that is linked from any other. This must use a UUID. Although these UUIDs don't technically have to be unique every time, in practice it is better to use newly generated ones, for each pairing of resources. Don't use the ones from examples. Replace all the referencing ids between resources with the same UUID as the relevant target fullUrl, so the links are preserved. Remove the actual ids of all resources, because these will be server assigned. POST the bundle.

e.g.

```
<Bundle xmlns="http://hl7.org/fhir">
... snip ...
   <fullUrl value="urn:uuid:b87c6544-bbc1-48f9-8cd5-39eaf7ccabf7"/>
   <MedicinalProductDefinition>
... snip ...
   <!-- remove id, gets a new one -->
   <RegulatedAuthorization>
       <!-- remove id, gets a new one -->
       <subject>
           <!-- to match above fullUrl -->
           <reference value="urn:uuid:b87c6544-bbc1-48f9-8cd5-39eaf7ccabf7"/>
       </subject>
```

Note that business identifiers within resources (using "identifier" elements) are different to ids, and references, and are not affected by this.

6.4.6. **(EP305) Get Product Part**

Use this operation to return information for a specific part of the data of a medicinal product, identified by its resource-id. This allows individual access to the linked components that go to make a whole medicinal product.

For the list of applicable resources see section 7.

The id used will typically be obtained from a reference within a MedicinalProductDefinition resource, or whichever resource is referencing the requested item.

Endpoint	GET /v{version}/{Product Part Resource Type}/{resource-id}
Request	
Accept	application/fhir+xml application/fhir+json
Body	n/a
Content-Type	n/a
Response	
Body	Resource of requested type

Path Parameters

Name	Description
resource-id	A unique resource identifier UUID **Example value:** a218e789-f917-48f5-8e2f-f64dc4ece8a4
version	Service version number **Example value: 2**

Query Parameters

None

Example Request

```
GET /v2/RegulatedAuthorization/4be6d0b5-9d39-4367-9c6d-ed030790db01
```

6.4.7. (EP306) Get Product Version

Use this operation to return an old version of specific product, identified by its product-id.

Resource Information

Endpoint	GET /v{version}/MedicinalProductDefinition/{product-id}/_history/{version-number}
Request	
Accept	application/fhir+xml application/fhir+json
Body	n/a
Content-Type	n/a
Response	
Body	MedicinalProductDefinition resource

Path Parameters

Name	Description
product-id	A unique product identifier UUID **Example value:** 4be6d0b5-9d39-4367-9c6d-ed030790db01
version-number	A version number **Example value:** 2
version	Service version number **Example value: 2**

Query Parameters

None

Example Request

```
GET /v2/MedicinalProductDefinition/4be6d0b5-9d39-4367-9c6d-ed030790db01/_history/2
```

6.4.8. (EP307) Get Product Versions

Use this operation to return all old versions of specific product, without needing to know the version ids of them. This only returns the MedicinalProductDefinition resource, not any of the link resources. Use other API calls to retrieve the linked parts.

Resource Information

Endpoint	GET /v{version}/MedicinalProductDefinition/{product-id}/_history
Request	

Accept	application/fhir+xml application/fhir+json
Body	n/a
Content-Type	n/a
Response	
Body	<Bundle of MedicinalProductDefinitions> e.g. Bundle entry MedicinalProductDefinition *

Path Parameters

Name	Description
product-id	A unique product identifier UUID **Example value:** 4be6d0b5-9d39-4367-9c6d-ed030790db01
version	Service version number **Example value: 2**

Query Parameters

None

Example Request

```
GET /v2/MedicinalProductDefinition/4be6d0b5-9d39-4367-9c6d-ed030790db01/_history
```

6.4.9. (EP306a) Get Product Version Full

Use this operation to return a complete old version of specific product, identified by its product-id, including all the old associated product parts, as they were at the time. This gets a history version of a MedicinalProductDefinition and the history versions of other resources that linked to it. Note that FHIR references between resources are not usually version specific. So, without this operation, the links in a history version of a resource will normally resolve the latest version of the linked resources – which may not be the version that existing at the relevant time. Although it is possible to reconstruct the original by querying for history versions of the linked resources by date, this operation is a much more convenient method.

Resource Information

Endpoint	GET /v{version}/MedicinalProductDefinition/{product-id}/$everything-history?version={version-number}
Request	
Accept	application/fhir+xml application/fhir+json
Body	n/a
Content-Type	n/a
Response	
Body	<Bundle of MedicinalProductDefinition and other types>

	e.g. Bundle entry MedicinalProductDefinition [entry {Product Part Resource Type}] *

Path Parameters

Name	Description
product-id	A unique product identifier UUID **Example value:** 4be6d0b5-9d39-4367-9c6d-ed030790db01
version-number	A version number **Example value:** 2
version	Service version number **Example value: 2**

Query Parameters

None

Example Request

```
GET /v2/MedicinalProductDefinition/4be6d0b5-9d39-4367-9c6d-ed030790db01/$everything-history?version=2
```

6.4.10. (EP308) Get Product Part Version

Use this operation to return an old version of specific product part, identified by its resource-id.

Resource Information

Endpoint	GET /v{version}/{Product Part Resource Type}/{resource-id}/_history/{version-number}
Request	
Accept	application/fhir+xml application/fhir+json
Body	n/a
Content-Type	n/a
Response	
Body	Resource of requested type

Path Parameters

Name	Description
resource-id	A unique resource identifier UUID

Name	Description
	Example value: a218e789-f917-48f5-8e2f-f64dc4ece8a4
version-number	A version number **Example value:** 2
version	Service version number **Example value: 2**

Query Parameters

None

Example Request

```
GET /v2/ClinicalUseIssue/4be6d0b5-9d39-4367-9c6d-ed030790db01/_history/2
```

6.4.11. (EP308a) Get Product Part Versions

Use this operation to return all the old versions of specific product part, identified by its resource-id, without needing to know the version ids of them.

Resource Information

Endpoint	GET /v{version}/{Product Part Resource Type}/{resource-id}/_history
Request	
Accept	application/fhir+xml application/fhir+json
Body	n/a
Content-Type	n/a
Response	
Body	<Bundle of versions of resource of requested type> e.g. Bundle entry {Product Part Resource Type} *

Path Parameters

Name	Description
resource-id	A unique resource identifier UUID **Example value:** a218e789-f917-48f5-8e2f-f64dc4ece8a4
version	Service version number **Example value: 2**

Query Parameters

None

Example Request

```
GET /v2/RegulatedAuthorization/5d681f34-3717-43cb-90d4-8a37e32c9710/_history
```

6.4.12. **(EP309) Create Product**

Use this operation to create a product, using a linked set of resources in a transaction bundle. The endpoint allows a product to be created as draft, so that it is kept in a working state before it is actually submitted (see EP315). In order to achieve that, set MedicinalProductDefinition.status as draft in the payload of the endpoint. In case this status is not set as draft, the product will be automatically submitted.

See http://hl7.org/fhir/http.html#transaction for general details of transactions, and section 5.7.1.

Resource Information

Endpoint	POST /v{version} (root of server for this version)
Request	
Accept	application/fhir+xml application/fhir+json
Body	<Bundle (type=transaction) of MedicinalProductDefinition and other types e.g. Bundle type=transaction entry MedicinalProductDefinition [marketingAuthorization reference value="AuthorizationUuid "/> (temporary local id)] * request method value=POST [entry fullUrl value="AuthorizationUuid" (matching temporary local id) RegulatedAuthorization request method value=POST] * [entry fullUrl value=TempUuid (another temporary local id) {other Medicinal Product type resources (not MedicinalProductDefinition itself) } request method value=POST] *
	application/fhir+xml application/fhir+json
Response	
Body	<Bundle (type=transaction-response)> e.g. Bundle type value="transaction-response entry response (states id of created resource) [entry

	response (for other linked child resources)] *

Path Parameters

Name	Description
version	Service version number **Example value: 2**

Query Parameters

None

Example Request

```
POST /v2
```

6.4.13. (EP310) Create Product Part

Use this operation to directly create a product part. For the list of applicable resources see section 6.4.

See http://hl7.org/fhir/http.html#transaction for general details of transactions

Resource Information

Endpoint	POST /v{version}/{Product Part Resource Type}
Request	
Accept	application/fhir+xml application/fhir+json
Body	{Product Part Resource Type}
	application/fhir+xml application/fhir+json
Response	
Body	Resource of same type as sent (echoed back). The http Location header gives the URL of the created resource.

Path Parameters

Name	Description
version	Service version number **Example value: 2**

Query Parameters

None

Example Request

```
POST /v2/ClinicalUseIssue
```

6.4.14. **(EP311) Update Product**

Use this operation to update a product, and any of its connected parts.

Resource Information

Endpoint	POST /v{version} (root of server for this version)
Request	
Accept	application/fhir+xml application/fhir+json
Body	<Bundle (type=transaction) of MedicinalProductDefinition and other types e.g. Bundle type=transaction entry MedicinalProductDefinition id={MedicinalProductDefinitionId} [{attribute of Medicinal Product Resource Type} reference={resourceId}] * request method=PUT [entry id=resourceId {Product Part Resource Type} request method=PUT] *
Content-Type	application/fhir+xml application/fhir+json
Response	
Body	<Bundle (type=transaction-response)> e.g. Bundle Type=transaction-response entry response (states id of created resource) [entry response (for other linked child resources)] *

Path Parameters

Name	Description
product-id	A unique product identifier UUID **Example value:** 4be6d0b5-9d39-4367-9c6d-ed030790db01
version	Service version number **Example value: 2**

Query Parameters

None

Example Request

```
POST /v2
```

6.4.15. (EP312) Update Product Part

Use this operation to update a part of a product. For the list of applicable resources see section 6.4.

Resource Information

Endpoint	POST /v{version}/{Product Part Resource Type}/{resource-id}
Request	
Accept	application/fhir+xml application/fhir+json
Body	{Product Part Resource Type}
Content-Type	application/fhir+xml application/fhir+json
Response	
Body	none

Path Parameters

Name	Description
resource-id	A unique resource identifier UUID **Example value:** a218e789-f917-48f5-8e2f-f64dc4ece8a4
version	Service version number **Example value: 2**

Query Parameters

None

Example Request

```
PUT /v2/ClinicalUseIssue/93d18516-7a3a-4863-812b-d3c5d9b70d61
```

6.4.16. (EP313) Delete Product

Use this operation to delete a draft product from the server. This is not applicable to the live product data. For live data, the product status must be changed (e.g. to "nullified") and the product updated (see 6.4.13.).

Resource Information

Endpoint	DELETE /v{version}/MedicinalProductDefinition/{product-id}
Request	
Accept	application/fhir+xml application/fhir+json
Body	none

Content-Type	application/fhir+xml application/fhir+json
Response	
Body	none

Path Parameters

Name	Description
product-id	A unique product identifier UUID **Example value:** 4be6d0b5-9d39-4367-9c6d-ed030790db01
version	Service version number **Example value: 2**

Query Parameters

None

Example Request

```
DELETE /v2/MedicinalProductDefinition/4be6d0b5-9d39-4367-9c6d-ed030790db01
```

6.4.17. (EP314) Delete Product Part

Use this operation to delete a draft product part from the server. This is not applicable to the live product data server. For the list of applicable resources see section 6.4.

Resource Information

Endpoint	DELETE /v{version}/{Product Part Resource Type}/{resource-id}
Request	
Accept	application/fhir+xml application/fhir+json
Body	none
Content-Type	application/fhir+xml application/fhir+json
Response	
Body	none

Path Parameters

Name	Description
resource-id	A unique resource identifier UUID **Example value:** a218e789-f917-48f5-8e2f-f64dc4ece8a4
version	Service version number **Example value: 2**

Query Parameters

None

Example Request

```
DELETE /v2/ ClinicalUseIssue/4be6d0b5-9d39-4367-9c6d-ed030790db01
```

6.4.18. (EP315) Submit Product

Use this operation to submit a draft product for approval. The status of the record will change to PENDING, but there are no other outwardly visible changes at this.

Resource Information

Endpoint	POST /v{version}/MedicinalProductDefinition/{product-id}/$submit
Request	
Accept	application/fhir+xml application/fhir+json
Body	n/a (no body, which is unusual for a post)
Content-Type	n/a
Response	
Body	MedicinalProductDefinition resource, echoed back, with the new status.

Path Parameters

Name	Description
product-id	A unique product identifier UUID **Example value:** 4be6d0b5-9d39-4367-9c6d-ed030790db01
version	Service version number **Example value: 2**

Query Parameters

None

Example Request

```
POST /v2/MedicinalProductDefinition/4be6d0b5-9d39-4367-9c6d-ed030790db01/$submit
```

6.4.19. (EP316) Create Draft version of existing Product

Use this operation to create a draft version of an existing a product on the server. This creates a new set of resources with the same content, but with different identifiers and allows the same functionality as described in EP309, only for an existing product. Therefore, the resulting version of the product will remain in draft state until it gets submitted. Optionally, a specific older version of the product may be duplicated. If no version parameter is specified, then the latest available version will be used.

Resource Information

Endpoint	POST /v{version}/MedicinalProductDefinition/{product-id}/$create-draft[?version={version-number}]
Request	
Accept	application/fhir+xml application/fhir+json
Body	n/a (no body, which is unusual for a post)
Content-Type	n/a
Response	
Body	A new MedicinalProductDefinition resource, echoed back. The http Location header gives the URL of the created resource.

Path Parameters

Name	Description
product-id	A unique product identifier UUID **Example value:** 4be6d0b5-9d39-4367-9c6d-ed030790db01
version-number	A version number **Example value:** 2
version	Service version number **Example value: 2**

Query Parameters

None

Example Request

```
POST /v2/MedicinalProductDefinition/4be6d0b5-9d39-4367-9c6d-ed030790db01/$create-draft
```

or

```
POST /v2/MedicinalProductDefinition/4be6d0b5-9d39-4367-9c6d-ed030790db01/$create-draft?version=2
```

6.4.20. **(EP317) Get Product Drafts**

Use this operation to find the draft copies of a Product. Draft versions can be accessed in the normal ways (e.g. see 6.4.4.) but this requires knowing their id. This operation will return all drafts linked to any version of a product. The ids can then be used to retrieve other details.

Resource Information

Endpoint	GET /v{version}/MedicinalProductDefinition/{product-id}/$drafts
Request	
Accept	application/fhir+xml application/fhir+json
Body	n/a
Content-Type	n/a

Response	
Body	<Bundle of MedicinalProductDefinitions> e.g. Bundle entry MedicinalProductDefinition *

Path Parameters

Name	Description
product-id	A unique product identifier UUID **Example value:** 4be6d0b5-9d39-4367-9c6d-ed030790db01
version	Service version number **Example value: 2**

Query Parameters

None

Example Request

```
GET /v2/MedicinalProductDefinition/4be6d0b5-9d39-4367-9c6d-ed030790db01/$drafts
```

6.4.21. (EP318) Validate Product

A validation interface is available for the product related resources. For the list of applicable resources see section 7.

6.5. *Substance Service*

This service enables the user to retrieve a single substance (SubstanceDefinition) via one of its identifiers, or a collection of substances based on provided search criteria. Direct creation or updating of substances is not allowed (See the Change Request SMS Service). Language translation of Substances is also possible.

6.5.1. Substance Versioning

Previous versions of substances are maintained. These can be accessed via the standard FHIR history mechanism (see http://hl7.org/fhir/http.html#history).

See also Get Substance Version 6.5.4. and Get Substance Versions 6.5.5.

6.5.2. (EP201) Search Substance

Use this operation to return information for specific substances, identified by search parameters. This operation supports server-side paging (see 5.9.).

Resource Information

Endpoint	GET /v{version}/SubstanceDefinition?{param}={value}[&{param}={value}]
Request	
Accept	application/fhir+xml application/fhir+json
Body	n/a
Content-Type	n/a
Response	
Body	Bundle of SubstanceDefinition(s) e.g. Bundle total=N [entry SubstanceDefinition] *

Path Parameters

Name	Description
version	Service version number **Example value: 2**

Query Parameters

Name	Description
identifier	Substance identifier
name	Substance name
domain	Human / Vet
code	Substance code
category	Substance type e.g. Chemical or Polymer
_lastUpdated	Searches on the modified date of the data e.g. _lastUpdated=ge2018-01-01&_lastUpdated=le2018-12-31
/v{version}/Provenance?_contained=true&when	Search on the declared updated date of the substance, rather than the record save time (which would be _lastUpdated). This search uses the Provenance endpoint, but searches the embedded ("contained") Provenance within the SubstanceDefinition. Use of "_contained=true" is necessary to allow searching this embedded data. The "target" parameter is then available, and represents the substance, to all for testing its other fields.

Name	Description
	Example: GET /v{version}/Provenance? _contained=true&when=gt2010-10-01&target.code=ABC123

Example Request

```
GET /v2/SubstanceDefinition?_lastUpdated=ge2018-01-01&_lastUpdated=le2018-12-31
```

6.5.3. (EP202) Get Substance

Use this operation to return information for a specific substance, identified by its substance-id.

Resource Information

Endpoint	GET /v{version}/SubstanceDefinition/{substance-id}
Request	
Accept	application/fhir+xml application/fhir+json
Body	n/a
Content-Type	n/a
Response	
Body	SubstanceDefinition

Path Parameters

Name	Description
substance-id	Unique substance identifier within SMS **Example value:** 4be6d0b5-9d39-4367-9c6d-ed030790db01
version	Service version number **Example value:** 2

Query Parameters

None

Example Request

```
GET /v2/SubstanceDefinition/4be6d0b5-9d39-4367-9c6d-ed030790db01
```

6.5.4. (EP203) Get Substance Version

Use this operation to return an old version of a specific substance, identified by its substance-id.

Resource Information

Endpoint	GET /v{version}/SubstanceDefinition/{substance-id}/_history/{version-number}
Request	
Accept	application/fhir+xml

	application/fhir+json
Body	n/a
Content-Type	n/a
Response	
Body	SubstanceDefinition

Path Parameters

Name	Description
substance-id	Unique substance identifier within SMS **Example value:** 4be6d0b5-9d39-4367-9c6d-ed030790db01
version-number	A version number Example value: 2
version	Service version number **Example value: 2**

Query Parameters

None

Example Request

```
GET /v2/SubstanceDefinition/4be6d0b5-9d39-4367-9c6d-ed030790db01/_history/2
```

6.5.5. (EP204) Get Substance Versions

Use this operation to return all old versions of specific substance, without needing to know the version ids of them.

Resource Information

Endpoint	GET /v{version}/SubstanceDefinition/{substance-id}/_history
Request	
Accept	application/fhir+xml application/fhir+json
Body	n/a
Content-Type	n/a
Response	
Body	<Bundle of SubstanceDefinitions> e.g. Bundle entry SubstanceDefinition *

Path Parameters

Name	Description
substance-id	Unique substance identifier within SMS **Example value:** 4be6d0b5-9d39-4367-9c6d-ed030790db01
version	Service version number **Example value: 2**

Query Parameters

None

Example Request

```
GET /v2/SubstanceDefinition/4be6d0b5-9d39-4367-9c6d-ed030790db01/_history
```

6.5.6. Create Substance

Not supported; In order to suggest new substance to be created, submit a new substance Change Request (of type create). See 6.7.3. (EP233) Create Change Request SMS

6.5.7. Update Substance

Not supported; In order to suggest update to a substance, submit a new substance Change Request (of type update)

See 6.7.3. (EP233) Create Change Request SMS

6.5.8. (EP207) Update Substance translations

This is a FHIR based method to bulk update the translations for a set of Substances (SubstanceDefinition resources). See also the file-based update here: 6.6.2.

This is an asynchronous service (initially returning only a 202 Accepted code, or a 40x series code, for failure). Subsequently, when processing has completed, a confirmation email shall be sent back to the submitting user.

This service does not work the same as a normal FHIR update (which is why it is an $ operation). It accepts "incomplete" SubstanceDefinition resources, having only an identifier and some names, with their translations. It then merges this content into the existing resources, rather than deleting all other substance fields that are not present in the uploaded data. However, existing translations for a given language and substance are removed. If any translations for a language for a given Substance name are included in the update, all other translations for that name and language will be deleted.

Endpoint	POST /v{version}/$putSubstanceTranslations (operates on server version root)
Request	
Accept	application/fhir+xml application/fhir+json
Body	<Bundle of (skeletal) SubstanceDefinitions (bundle type=batch)> e.g. Bundle

	type=batch [entry SubstanceDefinition (with only identifier and names/translations)] *
Content-Type	application/fhir+xml application/fhir+json
Response	
Body	n/a, or an OperationOutcome, in case of immediate failure.

6.5.9. (EP208) Validate Substance

A validation interface is available for the substance resources, SubstanceDefinition. See section 7.8.

Note that SubstanceDefinition resources can be validated but are normally created or updated only via the Change Request SMS Service.

6.6. *Substance Translation Service*

This service enables the user to create and update Translations for substances and to view Translations that they have previously created, via a file-based interface.

Note that the translations of an individual substance are always automatically available when a SubstanceDefinition resource is fetched (e.g. via 6.5.3.).

6.6.1. (EP221) Get values for translatable substance attributes

Use this operation to return a file of translatable names for substances.

Resource Information

Endpoint	GET /v{version}/Binary/$getSubstanceTranslations[?language={language}][&substance-id={substance-id}]
Request	
Accept	application/vnd.ms-excel
Body	n/a
Content-Type	n/a
Response	
Body	An Excel file with the list of translations, having columns: Identifier, Name, Translation, Language

Path Parameters

Name	Description
version	Service version number **Example value:** 2

Query Parameters

Name	Description

Name	Description
substance-id	Unique substance identifier **Example value:** 100000000102
language	2 letter ISO language code followed by 2 letter ISO country code (if appropriate) Include translatable attributes in the specified language-country. Possible value: (default en if not provided) • a specific language (i.e. fr or fr-ca). If the language is different than en, en would also be included implicitly **Example values:** en-us, de-de

Example Request

```
GET /v2/Binary/$getSubstanceTranslations?language=fr-BE&substance-id=100000000102
```

6.6.2. (EP222) Update the translatable substance attributes for a specific substance and language/country

Use this operation to update the translatable substance attributes for a specific substance and language/country, with a file-based interface. This is an asynchronous service. The file will be accepted immediately (and a 202 Accepted returned, or 400 in case of error), but the actual translations may take a while to process. Subsequently, when processing has completed, a confirmation email shall be sent back to the submitting user.

When substance is updated with a set of translations in a given language, all other translations in that language are removed. English names can only by updated by EMA users.

Hence, for any combination of substance identifier and language that appears in the file, the set of translated names is set to be the set of names in the file. Any others in that language, that are not mentioned in the file, will be removed.

Resource Information

Endpoint	POST /v{version}/Binary/$putSubstanceTranslations
Request	
Accept	application/fhir+xml application/fhir+json
Body	Excel file data, with columns Identifier, Name, Translation, Language
Content-Type	application/vnd.ms-excel
Response	
Body	n/a or OperationOutcome in case of immediate failure

Path Parameters

Name	Description
version	Service version number **Example value: 2**

The contents of the file indicate which substances are to be updated, and which are the desired translations in the languages included.

Example Request

```
PUT /v2/Binary/$putSubstanceTranslations
```

6.7. *Change Request SMS Service*

This service enables the user to create a new change request to request the creation/update of a substance (via a SubstanceDefinition resource).

Services are not provided to create/update/delete SubstanceDefinitions directly. All such operations are facilitated via change requests services. A change request is represented by the FHIR Task resource. A FHIR bundle of two resources, Task, and a SubstanceDefinition that the Task refers to, is the way to communicate an SMS change request.

Each change request must include a Request Reason. This informs the Data Steward the type of change being requested. The justification attribute can be used to provide more information about the reason for the change request, and to provide any supplementary information.

This service also enables the user to retrieve a single change request via one of its identifiers, or a collection of all the change requests that the user has created. A user is only able to retrieve their own change requests; they are not able to retrieve change requests raised by other users.

This service enables the user to create and update SMS Change Requests, to view SMS Change Requests that they have previously created, or to remove any SMS Change Requests they no longer want.

6.7.1. **(EP231) Search Change Requests SMS**

Use this operation to return a collection of change-requests, based on provided search criteria. This operation supports server-side paging (see 5.8.). This also supports returning the linked SubstanceDefinitions, using the "_include" parameter.

Resource Information

Endpoint	GET /v{version}/Task?{param}={value}[&{param}={value}]
Request	
Accept	application/fhir+xml application/fhir+json
Body	n/a
Content-Type	n/a
Response	
Body	Bundle of Tasks and optionally SubstanceDefinitions (if _included) e.g. Bundle total=N [entry Task] * [entry SubstanceDefinition] *

Path Parameters

Name	Description
version	Service version number **Example value: 2**

Query Parameters

Name	Description
code:text	Change Request name **Example value:** RoAdm-Other%20use%20 request (note use of "%20" to stand in for spaces. No quotes are used around strings)
identifier	Change Request identifier **Example value:** RRQ-1000000002
businessStatus	Search only for change requests with these statuses. If this parameter is not provided, search all change requests. Possible values[2]: *SAVED* *SUBMITTED* *VALID* *INVALID* *RETURNED* *APPROVED* *APPROVED_WC* *REJECTED* blank (default value) **Example value:** SUBMITTED,SAVED
intent	Search only for change requests with these types. If this parameter is not provided, search all change requests. Possible values: blank (default value) *ADD_SUBSTANCE* *UPD_SUBSTANCE* *DEL_SUBSTANCE* **Example value:** *ADD-SUBSTANCE*
authored-on	Format = YYYY-MM-DDThh:mm:ssZ e.g. ?date=2016-05-09T11:58:00Z Search on before or after using =lt or =gt e.g. ?date=gt2016-05-09T11:58:00Z **Example value:** 2016-05-09T11:58:00
focus:SubstanceDefinition.name	Substance name in SMS

[2] These values are defined in the SPOR glossary of terms, which is published as a separate document.

Name	Description
	Example value: Rotherdam
focus:identfier	Substance id in SMS
_include	fixed value "Task:focus" e.g. _include=Task:focus This causes the linked SubstanceDefinition to also be returned in the results bundle.

Example Request

```
GET /v2/task?name=republic&businessStatus=SUBMITTED,REJECTED
```

6.7.2. (EP232) Get Change Request SMS

Use this operation to return information for a specific change request, identified by its change-request-id. Note that this only returns the Task itself and not the associated substance. The Task contains a reference to the SubstanceDefinition that can be used to then fetch the substance, using Get Substance 6.5.3.

Alternatively, the Search Change Requests SMS operation (6.7.1.) can find the task and also fetch the associated SubstanceDefinition.

Resource Information

Endpoint	GET /v{version}/Task/{change-request-id}
Request	
Accept	application/fhir+xml application/fhir+json
Body	n/a
Content-Type	n/a
Response	
Body	Task

Path Parameters

Name	Description
change-request-id	Change Request Identifier **Example value:** RRQ-100000123
version	Service version number **Example value: 2**

Query Parameters

None

Example Request

```
GET /v2/Task/RRQ-100000123
```

6.7.3. (EP233) Create Change Request SMS

Use this operation to create a new change request.

Change requests use a Task resource and optionally a SubstanceDefinition resource (for "add substance", or "update substance", but not for "delete substance").

For "delete substance" change requests, an alternative/replacement substance(s) may be specified for the substance being deleted.

Supporting documents that are associated with the substance use the DocumentReference resource (as a "contained" resource within the SubstanceDefinition). The DocumentReference contains the URL of the document. This document must be uploaded beforehand using the V1 API for Documents. See section 9.1.

Note: Services are not provided to create/update/delete substances directly. All such operations are facilitated via change requests services.

Resource Information

Endpoint	POST /v{version} (the root of the server for this version)
Request	
Accept	application/fhir+xml application/fhir+json
Body	Bundle (type=transaction) of Task (type=cr-type, id=<not to be specified>, subject=) SubstanceDefinition.fullUrl), and a SubstanceDefinition (fullUrl is a UUID, which is used as Task.subject) e.g. Bundle type=transaction entry Task [subject (needed except for a delete) reference={SubstanceUuid} (temporary client created id) {other task fields etc.}] method=POST [entry (needed except for a delete) fullUrl={SubstanceUuid} (matching temporary local id) SubstanceDefinition [contained DocumentReference url={document from V1 api}] * {other substance fields etc.} method=POST] *
	application/fhir+xml application/fhir+json
Response	
Body	Bundle (type=transaction-response) with the id of the created Task and the id of the SubstanceDefinition. The SubstanceDefinition id can be considered a temporary id, for use until the SubstanceDefinition is accepted. Bundle type=transaction-response entry response (states id of created Task) location={URL of task, including id} [entry (except when deleting a substance) response (states temp id of created SubstanceDefinition) location={URL of SubstanceDefinition, including id}]

Path Parameters

Name	Description
version	Service version number **Example value: 2**

Query Parameters

none

Example Request

```
POST /v2
```

6.7.4. (EP234) Update Change Request SMS

Use this operation to update an existing change request. Note that this is an update to an existing change request, and not a change request for an update to a substance. The latter would be a new change request, using 6.7.3. .

A change request may be of the following types: "add substance", "update substance ", "delete substance".

An update on any type of change request must contain its request-id. Otherwise the change request will be rejected.

Change requests relating to substances (i.e. "add substance", "update substance", "delete substance") have change request attributes plus optional details of associated documents and controlled substance. When an update on an already created change request relating to substances is executed, the relevant draft-substance identifiers previously obtained in the change request creation must be retained, that is, provided again. Any new element within the change request must render no identifier. However, once the response is obtained, the new identifiers assigned by the system to those elements must also be stored, in order to provide them back in the case of a new change request update.

For "delete substance" change requests an alternative/replacement substance(s) may be specified for the substance being deleted.

Note: Documents associated with change requests are uploaded using the V1 service (see section 9.1.1.). The SubstanceDefinition resource in this update can point to the same or different document RLs.

Note: Services are not provided to create/update/delete Lists and Substances directly. All such operations are facilitated via change requests services.

Resource Information

Endpoint	PUT /v{version}/Task/{change-request-id}
Request	
Accept	application/fhir+xml application/fhir+json
Body	Bundle (type=transaction) of Task (type=cr-type, id=<not to be specified>, subject=) SubstanceDefinition.fullUrl), and a SubstanceDefinition (fullUrl is a

	UUID, which is used as Task.subject) e.g. Bundle type=transaction entry Task [subject (needed except for a delete) Reference={SubstanceUuid} (temporary client created id) {other task fields etc.}] method=PUT [entry (needed except for a delete) fullUrl={SubstanceUuid} (matching temporary local id) SubstanceDefinition [contained DocumentReference url={document from V1 api}] * {other substance fields etc.} method=PUT] *
Content-Type	application/fhir+xml application/fhir+json
Response	
Body	n/a

Path Parameters

Name	Description
change-request-id	Change Request Identifier **Example value:** RRQ-100000123
version	Service version number **Example value: 2**

Query Parameters

None

Example Request

```
PUT /v2/Task/RRQ-100000123
```

6.7.5. **(EP235) Delete Change Request SMS**

Use this operation to delete an existing change request. Note that this is a delete of a request, and not a request to delete a substance, or a direct delete of a substance.

Deletion is only allowed if the Change Request has status 'Saved'. Once the change request is submitted (status is different than 'Saved'), the change request cannot be deleted.

"Status" refers to the where the Change Request is in the saved/submitted/approve|rejectlife-cycle

Resource Information

Endpoint	DELETE /v{version}/Task/{change-request-id}

Request	
Accept	n/a
Body	n/a
Content-Type	n/a
Response	
Body	n/a

Path Parameters

Name	Description
change-request-id	Unique identifier of change request Example value: RRQ-1000000001
version	Service version number **Example value: 2**

Example Request

```
DELETE /v2/Task/RRQ-1000000001
```

6.7.6. (EP236) Validate Change Request

A validation interface is available for the change request resource, Task. See section 6.3.

7. Resources

Except where stated, all resources are as modelled as a FHIR resources as documented here: http://build.fhir.org/resourcelist.html

Extensions and special cases are documented below.

7.1. *MedicinalProductDefinition*

http://build.fhir.org/medicinalproductdefinition.html

SPOR API v2 implementation of this resource requires additional extensions:

Extension:

Canonical URL	http://ema.europa.eu/fhir/extension/versionBasedOn
Type	String
Extends	MedicinalProductDefinition
LDM item	MedicinalProduct/VersionBasedOn

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<MedicinalProductDefinition xmlns="http://hl7.org/fhir">
    <extension url="http://ema.europa.eu/fhir/extension/versionBasedOn"/>
       <valueString value="d08ee2ff-af11-47cd-9049-0f5c9320df27"/>
    </extension>
```

Extension:

Canonical URL	http://ema.europa.eu/fhir/extension/atcPending
Type	Boolean
Extends	MedicinalProductDefinition
LDM item	MedicinalProduct/ATC Application Flag

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<MedicinalProductDefinition xmlns="http://hl7.org/fhir">
    <extension url="http://ema.europa.eu/fhir/extension/atcPending "/>
       <valueBoolean value="false"/>
    </extension>
```

7.2. *RegulatedAuthorization*

http://build.fhir.org/regulatedauthorization.html

SPOR API v2 implementation of this resource requires additional extensions:

Extension:

Canonical URL	http://ema.europa.eu/fhir/extension/targetSpecies
Type	Coding
Extends	RegulatedAuthorization
LDM item	MedicinalProduct/RegulatoryEntitlement/TargetSpecies

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<RegulatedAuthorization xmlns="http://hl7.org/fhir">
    <extension url="http://ema.europa.eu/fhir/extension/targetSpecies"/>
       <valueCoding>
           <system value="http://ema.europa.eu/example/species"/>
           <code value="bovine"/>
       </valueCoding>
   </extension>
```

Extension:

Canonical URL	http://ema.europa.eu/fhir/extension/indication
Type	String

Extends	RegulatedAuthorization
LDM item	MedicinalProduct/Regulatory Entitlement/Minor Use Minor Species/MUMS Indication Text

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<RegulatedAuthorization xmlns="http://hl7.org/fhir">
    <extension url="http://ema.europa.eu/fhir/extension/indication"/>
       <valueString value="Symptomatic treatment of … "/>
   </extension>
```

Extension:

Canonical URL	http://ema.europa.eu/fhir/extension/parallelTradeWholesaler
Type	Coding
Extends	RegulatedAuthorization
LDM item	MedicinalProduct/RegulatoryEntitlement/ParallelTradeWholesaler

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<RegulatedAuthorization xmlns="http://hl7.org/fhir">
    <extension url="http://ema.europa.eu/fhir/extension/parallelTradeWholesaler"/>
       <valueCoding>
           <system value="http://ema.europa.eu/example/locations"/>
           <code value="LOC-1234567890"/>
       </valueCoding>
    </extension>
```

Extension:

Canonical URL	http://ema.europa.eu/fhir/extension/referenceCountry
Type	Coding
Extends	RegulatedAuthorization.case
LDM item	MedicinalProduct/RegulatoryProcedure/ReferenceCountry

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<RegulatedAuthorization xmlns="http://hl7.org/fhir">
   <case>
       <extension url="http://ema.europa.eu/fhir/extension/referenceCountry "/>
          <valueCoding>
              <system value="http://ema.europa.eu/example/countries"/>
              <code value="Spain"/>
          </valueCoding>
       </extension>
    </case>
```

Extension:

Canonical URL	http://ema.europa.eu/fhir/extension/concernedCountries
Type	Coding
Extends	RegulatedAuthorization.case
LDM item	MedicinalProduct/RegulatoryProcedure/ConcernedCountries

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<RegulatedAuthorization xmlns="http://hl7.org/fhir">
    <case>
       <extension url="http://ema.europa.eu/fhir/extension/concernedCountries"/>
          <valueCoding>
              <system value="http://ema.europa.eu/example/countries"/>
              <code value="Italy"/>
          </valueCoding>
          <valueCoding>
              <system value="http://ema.europa.eu/example/countries"/>
              <code value="Poland"/>
          </valueCoding>
       </extension>
    </case>
```

Extension:

Canonical URL	http://ema.europa.eu/fhir/extension/variationClassification
Type	Coding
Extends	RegulatedAuthorization.case.application
LDM item	MedicinalProduct/RegulatoryProcedure/ VariationNotRequiringScientificAssessment/VariationClassification

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<RegulatedAuthorization xmlns="http://hl7.org/fhir">
   <case>
       <application>
          <extension url="http://ema.europa.eu/fhir/extension/variationClassification"/>
             <valueCoding>
                 <system value="http://ema.europa.eu/example/variationClassifications"/>
                 <code value="A.1 Change in the name and/or address of the marketing
authorisation holder"/>
             </valueCoding>
          </extension>
       </application>
    </case>
```

Extension:

Canonical URL	http://ema.europa.eu/fhir/extension/submissionComment
Type	String
Extends	RegulatedAuthorization.case.application
LDM item	MedicinalProduct/RegulatoryProcedure/ VariationNotRequiringScientificAssessment/SubmissionComment

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<RegulatedAuthorization xmlns="http://hl7.org/fhir">
   <case>
       <application>
          <extension url="http://ema.europa.eu/fhir/extension/submissionComment"/>
             <valueString value="This is a comment"/>
          </extension>
       </application>
    </case>
```

Extension:

Canonical URL	http://ema.europa.eu/fhir/extension/authorOfDecision
Type	String
Extends	RegulatedAuthorization.case.application
LDM item	MedicinalProduct/RegulatoryProcedure/ VariationNotRequiringScientificAssessment/AuthorOfDecision

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<RegulatedAuthorization xmlns="http://hl7.org/fhir">
   <case>
       <application>
          <extension url="http://ema.europa.eu/fhir/extension/authorOfDecision"/>
             <valueString value="This is a comment"/>
          </extension>
       </application>
    </case>
```

7.3. *ClinicalUseIssue*

http://build.fhir.org/clinicaluseissue.html

SPOR API v2 implementation of this resource requires additional extensions:

Extension:

Canonical URL	http://ema.europa.eu/fhir/extension/targetSpecies

Type	Coding
Extends	ClinicalUseIssue
LDM item	MedicinalProduct/Contraindication/TherapeuticIndication/TargetSpecies

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<ClinicalUseIssue xmlns="http://hl7.org/fhir">
    <extension url="http://ema.europa.eu/fhir/extension/targetSpecies"/>
       <valueCoding>
           <system value="http://ema.europa.eu/example/species"/>
           <code value="bovine"/>
       </valueCoding>
   </extension>
```

7.4. ***Ingredient***

http://build.fhir.org/ingredient.html

Extension:

Canonical URL	http://ema.europa.eu/fhir/extension/presentationStrengthText
Type	String
Extends	Ingredient/strength
LDM item	MedicinalProduct/PharmaceuticalProduct/Ingredient/Ingredient Substance Strength/Presentation Strength Text

Extension:

Canonical URL	http://ema.europa.eu/fhir/extension/concentrationStrengthText
Type	String
Extends	Ingredient/strength
LDM item	MedicinalProduct/PharmaceuticalProduct/Ingredient/Ingredient Substance Strength/Concentration Strength Text

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<Ingredient xmlns="http://hl7.org/fhir">
...
    <strength>
       <extension url="http://ema.europa.eu/fhir/extension/presentationStrengthText"/>
              <valueString value="3,1 g/5 ml to 3,7 g/5 ml"/>
       </extension>
       <extension url="http://ema.europa.eu/fhir/extension/concentrationStrengthText"/>
              <valueString value="2mg/1tablet"/>
       </extension>
```

7.5. *PackagedProductDefinition*

http://build.fhir.org/packagedproductdefinition.html

7.6. *AdministrableProductDefinition*

http://build.fhir.org/administrableproductdefinition.html

Extension:

Canonical URL	http://ema.europa.eu/fhir/extension/maximumResidueLimit
Type	Complex (Coding and String)
Extends	AdministrableProductDefinition/RouteOfAdministration/TargetSpecies
LDM item	MedicinalProduct/PharmaceuticalProduct/RouteOfAdministration/TargetSpecies/MaximumResidueLimit

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<AdministrableProductDefinition xmlns="http://hl7.org/fhir">
...
    <routeOfAdministration>
       <targetSpecies>
           <extension url="http://ema.europa.eu/fhir/extension/maximumResidueLimit"/>
               <extension
url="http://ema.europa.eu/fhir/extension/maximumResidueLimitTissue"/>
                   <valueCoding>
                       <system value="http://ema.europa.eu/example/species"/>
                       <code value="bovine"/>
                   </valueCoding>
               </extension>
               <extension
url="http://ema.europa.eu/fhir/extension/maximumResidueDecisionNumber"/>
                   <valueString value="1234"/>
               </extension>
           </extension>
```

7.7. *ManufacturedItemDefinition*

http://build.fhir.org/manufactureditemdefinition.html

7.8. *SubstanceDefinition*

http://build.fhir.org/substancedefinition.html

Extension:

Canonical URL	http://ema.europa.eu/fhir/extension/dataClassification

Type	Coding
Extends	SubstanceDefinition SubstanceDefinition.Name
LDM item	Substance/Data Classification Term Id Substance Name/Data Classification Term Id

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<SubstanceDefinition xmlns="http://hl7.org/fhir">
       <extension url="http://ema.europa.eu/fhir/extension/dataClassification"/>
           <valueCoding>
               <system value="http://ema.europa.eu/example/dataClassification"/>
               <code value="PUBLIC"/>
           </valueCoding>
       </extension>
```

7.9. *DeviceDefinition*

http://build.fhir.org/devicedefinition.html

7.10. *Task*

http://www.hl7.org/fhir/R4/task.html

7.11. *DocumentReference*

http://www.hl7.org/fhir/R4/documentreference.html

SPOR API v2 implementation of this resource requires additional extensions:

Extension:

Canonical URL	http://ema.europa.eu/fhir/extension/documentLanguage
Type	Coding
Extends	DocumentReference/Content
LDM item	MedicinalProduct/AttachedDocument/Language

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<DocumentReference xmlns="http://hl7.org/fhir">
    <content>
       <extension url="http://ema.europa.eu/fhir/extension/documentLanguage"/>
```

```
            <valueCoding>
                <system value="http://ema.europa.eu/example/language"/>
                <code value="fr-BE"/>
            </valueCoding>
        </extension>
```

Extension:

Canonical URL	http://ema.europa.eu/fhir/extension/characterSet
Type	Coding
Extends	DocumentReference
LDM item	MedicinalProduct/AttachedDocument/Char_Set

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<DocumentReference xmlns="http://hl7.org/fhir">
    <extension url="http://ema.europa.eu/fhir/extension/characterSet"/>
       <valueCoding>
           <system value="http://ema.europa.eu/example/characterSet"/>
           <code value="UTF-8"/>
       </valueCoding>
   </extension>
```

Extension:

Canonical URL	http://ema.europa.eu/fhir/extension/fileName
Type	String
Extends	DocumentReference
LDM item	MedicinalProduct/AttachedDocument/FileName

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<DocumentReference xmlns="http://hl7.org/fhir">
    <extension url="http://ema.europa.eu/fhir/extension/fileName"/>
       <valueString value="example.pdf"/>
    </extension>
```

7.12. ***Bundle***

http://www.hl7.org/fhir/R4/bundle.htm

and see 5.7.

7.13. *OperationOutcome*

Used in the return from HTTP calls to document errors or warnings.

http://www.hl7.org/fhir/R4/operationoutcome.html

7.14. *CapabilityStatement*

A Capability Statement documents a set of capabilities (behaviours) of a FHIR Server. These will not be exchanged, but the server will expose a read-only CapabilityStatement describing its properties.

http://www.hl7.org/fhir/R4/capabilitystatement.html

8. About this Document

8.1. *Definitions, Acronyms, and Abbreviations*

Acronym/Abbreviation	Description
API	**Application Programming Interface**
EUTCT	**European Union Telematics Controlled Terms**
FHIR	**Fast Healthcare Interoperability Resources**
HTTP	**Hypertext Transfer Protocol**
IANA	**Internet Assigned Numbers Authority**
JSON	**JavaScript Object Notation**
MAH	**Marketing Authorisation Holder**
SPOR	**Substances, Products, Organisations, Referentials**
SMS	**Substances Management System**
PMS	**Products Management System**
OMS	**Organisations Management System**
RBAC	**Role Based Access Control**
REST	**Representational State Transfer**
RMS	**Referentials Management System**
TOM	**Target Operating Model**
UUID	**Universal Unique Identifier**
XML	**Extensible Mark-up Language**

8.2. *Open Issues*

- None

8.3. *Referenced documents*

Doc ID	Title	Location
EMA/241514/2016	SPOR-API-Specification (V1)	https://docs.eudra.org/webtop/drl/objectId/090142b283f11d52

8.4. *Document Approval*

Date	Version	Submitted by	Approved by	Approve role
18th Oct. 2018	1.0	Gustavo Rodriguez Rik Smithies	EAB	EAB

Date	Version	Submitted by	Approved by	Approve role
		Ray Power		
7th Nov. 2018	1.0	EMA To TEAB		

8.5. *Document history*

Version	Who	What
1.0	EMA	Creation for consultation
1.1	EMA	Added back references to MedicinalProductPharmaceutical resource
1.2	EMA	Updates from public consultation and FHIR maturity level 1 approval process
1.3	EMA	Search fields updated according to latest requirements

9. Annexes

9.1. *Annex I –Using SPOR V1 resources/ endpoints from SPOR V2*

This section describes the steps needed to use a SPOR API V1 resource from SPOR API V2,

9.1.1. Using the Document Service

This section describes the steps needed to use the SPOR API V1 "(EP51) Create Document" endpoint to create (upload) a new document. The method allows the content of the file to be uploaded as base64 and creates the file and its' metadata. The newly created (uploaded) document is then link to a MedicinalProductDefinition record.

1. Create a new "Document" object containing a base 64 representation of the new PMS document and associated meta-data. (The "Document" type is described in Section 7.5 of SPOR API V1 specification.)
2. Using the "Document" object as the body , `POST /v1/documents?app-domain=PMS&doc-type=GENERAL`
3. The response will contain a "Document" object populated with new Document ID e.g. 123
4. This document can be retrieved at any point using `GET /v1/document/123?app-domain=PMS&doc-type=GENERAL` and will be used to link the document to MedicinalProductDefinition resource. Using the URL reference, create a FHIR Attachment (http://www.hl7.org/fhir/R4/datatypes.html#Attachment)
5. Using the FHIR Attachment, create a FHIR DocumentReference object (http://www.hl7.org/fhir/R4/documentreference.html). "A DocumentReference resource is used to describe a document that is made available to a healthcare system".
6. Create a MedicinalProductDefinition object (http://build.fhir.org/medicinalproductdefinition.html) and add the previously created DocumentReference object.

7. This MedicinalProductDefinition object is then included in a FHIR a transaction bundle, in addition to other Product Parts, and POSTed to the server as described in Section (EP309) Create Product of this document.

9.1.2. **Using the SearchQuery Service**

This SearchQuery Service enables users to create a new SearchQuery (search criteria), to view a SearchQuery that they have previously created and update it and to remove a SearchQuery they no longer want.

This section describes the steps needed to use the SPOR API V1 endpoints to create a new search query, to return all search queries that the user has created and to return information for a specific search query, identified by its search-query-id.

1. Create a new "SearchQuery" object containing a representation of the query and associated meta-data. (The "SearchQuery" type is described in Section 7.6 of SPOR API V1 specification.)
2. Using the "SearchQuery" object as the body , `POST /v1/search-queries`
3. The response will contain a "SearchQuery" object populated with new SearchQuery ID e.g. 123
4. This SearchQuery can be retrieved at any point using `GET /v1/search-queries/123`
5. The query representation can be extracted from SearchQuery and used to create the search criteria for a Product (or Substance) search e.g. `GET /v{version}/MedicinalProductDefinition?{param}={value}[&{param}={value}]`

9.2. ***Annex II – Record versioning in SMS and PMS***

SMS and PMS are implemented based on a commercial master data management solution (MDM), which comes with several off the shelf features for managing data. One of them is keeping a full track of all the changes made to the mastered data and ability to reconstruct the state of a data record at any time point during this record lifetime.

This model assumes that data record versions are based on timestamps and makes it possible to query a version of a record at any timestamp in the past. This approach works naturally with other important MDM feature, which is merging of separate records into one master record. For example, if two separate records A and B got merged into a new master record AB at a timestamp T1, it is relatively easy to query their different versions based on timestamps: all queries for versions of the record A before the timestamp T1 will return an appropriate version of record A; but all queries for versions of the record A after the timestamp T1 will return an appropriate version of the merged record AB.

In addition to timestamp based versions, SMS and PMS also offers versions based on numeric values (e.g. v. 1, v. 2, v. 3, etc.) whereby any changes to a Substance or Product results in an incremented version number. This is mainly for backward compatibility with earlier systems (EUTCT & EV).

See Section 5.12. Metadata which describes how the API uses the FHIR implementation of versioning ("versionId") and timestamping ("lastUpdated").

When referencing any record in SMS and PMS, a client has a number of options:

- Reference only record identifier (i.e. Substance identifier or Product identifier) – in this case any call to SMS or PMS will always return the latest (visible) version of the referenced term. (For further detail on Product version visibility, see section 9.2.1. Latest (Visible) Product Version)

- Reference a record identifier (i.e. Substance identifier or Product identifier) and a timestamp when the reference was made. Thanks to MDM features described above it will always be possible to extract the exact version of referenced record at this timestamp.
- Reference a record identifier (i.e. Substance identifier or Product identifier) and its version number.

It is up to the client implementation and its use case and business process to decide which model of referencing SMS and PMS records is used.

9.2.1. **Latest (Visible) Product Version**

The "Latest Product Version" is the latest version of the "Product" record. This is always the most recent version of information which is stored in the System and is independent of items such as:

- "Product Status" information
- User access permissions
- User organisations

"Latest Public Product Version"

This is the latest version of the "Product" record which is a "Current Product Version".

"Provisional Product Version" information and "Nullified Product Version" information are not considered when identifying the "Latest Public Product Version".

"Provisional Product Version" information and "Nullified Product Version" information is not visible by the public, only users who are associated with the MAH for the "Product" are able to see "Provisional Product Version" information.

"Latest Visible Product Version"

This is the latest version of the "Product" information which the user has access control permissions to view. For a given "Product" record, different users could see different "Latest Visible Product Version" information based on the access control settings.

Example 1

Product Name = "abc101"

v1 - Provisional

v2 - Current

v3 - Provisional

v4 - Current

v5 - Provisional

v6 - Current

v7 - Provisional

v8 - Current

The "Latest Product Version" = v8.

The "Latest Public Product Version" = v8

The “Latest Visible Product Version” for a user associated with the MAH for the “Product” = v8

The “Latest Visible Product Version” for a user not associated with the MAH for the “Product” = v8

Example 2

Product Name = “abc102”

v1 - Provisional

v2 - Current

v3 - Provisional

v4 - Current

v5 - Provisional

v6 - Current

v7 - Provisional

v8 - Current

v9 - Provisional

v10 - Provisional

The “Latest Product Version” = v10.

The “Latest Public Product Version” = v8.

The “Latest Visible Product Version” for a user associated with the MAH for the “Product” = v10

The “Latest Visible Product Version” for a user not associated with the MAH for the “Product” = v8

Example 3

Product Name = “abc103”

v1 - Provisional

v2 - Provisional

v3 - Provisional

v4 - Provisional

v5 - Provisional

The “Latest Product Version” = v5.

The “Latest Public Product Version” = N/A – “Product” information not visible.

The “Latest Visible Product Version” for a user associated with the MAH for the “Product” = v5

The “Latest Visible Product Version” for a user not associated with the MAH for the “Product” = N/A – “Product” information not visible.

Example 4

Product Name = “abc104”

v1 - Nullified

v2 - Nullified

v3 - Nullified

v4 - Nullified

v5 - Nullified

The "Latest Product Version" = v5.

The "Latest Public Product Version" = N/A – "Product" information not visible.

The "Latest Visible Product Version" for a user associated with the MAH for the "Product" = v5

The "Latest Visible Product Version" for a user not associated with the MAH for the "Product" = N/A – "Product" information not visible.

The "Latest Visible Product Version" also takes "Draft Product" information into account by either including or excluding the "Draft Product" information, see examples 5 to 6.

Example 5

Product Name = "abc105"

v1 - Provisional

v2 - Current

v3 - Provisional

- "Draft Product" #1 which was had a base "Product Version" from v3.
- "Draft Product" #2 which was had a base "Product Version" from v3.

v4 - Current

v5 - Provisional

v6 - Current

v7 - Provisional

v8 - Current

- "Draft Product" which was had a base "Product Version" from v8.

v9 - Provisional

v10 - Provisional

- "Draft Product" which was had a base "Product Version" from v10.

The "Latest Product Version" = v10; the "Draft Product" from v10 is not taken into account.

The "Latest Public Product Version" = v8; the "Draft Product" from v8 is not taken into account.

The "Latest Visible Product Version" for a user associated with the MAH for the "Product" = v10; the "Draft Product" from v10 is not taken into account.

The "Latest Visible Product Version" for a user not associated with the MAH for the "Product" = v8; the "Draft Product" from v8 is not taken into account.

Example 6

Product Name = "abc106"

"Draft Product" exists for "abc106", but has never been successfully Submitted; therefore no "Product Version" information exists for "abc106".

The "Latest Product Version" = N/A – the "Draft Product" (with no "Product Version") is not taken into account.

The "Latest Public Product Version" N/A – the "Draft Product" (with no "Product Version") is not taken into account.

The "Latest Visible Product Version" for a user associated with the MAH for the "Product" = the "Draft Product" as no "Product Version" record is available.

The "Latest Visible Product Version" for a user not associated with the MAH for the "Product" = N/A – the "Draft Product" (with no "Product Version") is not taken into account.

9.3. *Annex III – SMS & PMS Synchronisation Mechanism*

It is possible to synchronise with SMS and PMS in two ways:

1. A client can use direct API calls directly to SMS/PMS to retrieve data and populate its local forms and data fields required data. This approach does require frequent calls to remote SMS/PMS API endpoints.
2. A client can make a local copy of relevant SMS/PMS data, and subsequently populate its local forms and data fields from this copy. This approach requires that the local copy periodically gets synchronised with the master data in SMS/PMS.

This section describes a synchronisation mechanism for the latter scenario.

1. The local copy of the SMS/PMS data has been created by copying the complete dataset at a timestamp T1.
2. A follow up synchronisation process is triggered by the client at the timestamp T2 to receive all the Substance / Product data that have been modified in SMS/PMS data in the period between timestamps T1 and T2.
3. Each subsequent synchronisation at the timestamp Tn+1 receives Substance / Product data modified since the timestamp Tn, n≥1.