



EUROPEAN MEDICINES AGENCY
SCIENCE MEDICINES HEALTH

1 1 July 2021
2 EMA/286879/2021
3 European Medicines Agency

4 **Draft Electronic Product Information (ePI) API**
5 **Specification**
6 **Version 1.1**

Start of public consultation	7 June 2021
End of consultation (deadline for comments)	31 July 2021

7

Comments should be provided using the online form:
<https://ec.europa.eu/eusurvey/runner/ePIStandardConsultation>

8



9	Table of contents	
10	1. Document purpose	4
11	2. Scope.....	4
12	3. Introduction	4
13	3.1. FHIR	4
14	3.2. Definitions.....	5
15	3.3. What the API is not	5
16	3.4. Flexibility and constraints	5
17	3.5. Spelling.....	5
18	4. Specification.....	6
19	4.1. Versioning	6
20	4.1.1. Versioning	6
21	4.1.2. XML schemas versioning.....	6
22	4.1.3. Service versioning	6
23	4.2. Authentication and authorisation	6
24	4.3. FHIR extensions	7
25	4.4. HTTP methods	7
26	4.5. HTTP errors and status.....	7
27	4.5.1. Asynchronous updates	8
28	4.6. FHIR references and identifiers	8
29	4.6.1. FHIR resource id (1..1).....	8
30	4.6.2. References.....	9
31	4.7. Bundles.....	10
32	4.7.1. Document bundles.....	11
33	4.7.2. Transaction bundles	11
34	4.7.3. Bundle endpoints.....	12
35	4.8. Searching	12
36	4.9. Paging and sorting.....	12
37	4.10. Resources and representations.....	13
38	4.11. Encoding	13
39	4.12. Request parameters and searches	13
40	4.12.1. Parameter characteristics.....	14
41	4.12.2. Full text search.....	14
42	4.12.3. Chained searches.....	14
43	4.12.4. Including other resources in search results	15
44	4.13. Metadata	15
45	4.14. Standards.....	16
46	5. Resource structure	16
47	5.1. Groups of resources.....	16
48	5.2. Scenarios and lifecycle	18
49	5.2.1. Scenario: creation with SmPC, package leaflet and 2 labelling entries	18
50	5.2.2. Scenario: creation of individual list entries.....	19
51	5.2.3. Scenario: adding a translation for the SmPC and labelling	19
52	5.2.4. Scenario: updating a package leaflet.....	20

53	5.2.5. Scenario: a new package leaflet and a labelling are added	21
54	5.2.6. Scenario: a labelling is deleted	22
55	5.2.7. Scenario: parallel distribution.....	22
56	6. REST services	23
57	6.1. Resource summary	23
58	6.2. Service summary	24
59	7. Resources.....	24
60	7.1. Bundle	24
61	7.2. List.....	24
62	7.3. Composition	25
63	7.4. Binary	25
64	7.5. MedicinalProductDefinition	25
65	7.6. OperationOutcome	26
66	7.7. CapabilityStatement	26
67	8. About this document	26
68	8.1. Definitions, acronyms, and abbreviations	26
69	8.2. Open issues.....	26
70	8.3. Document approval	26
71	8.4. Document history.....	27
72		

1. Document purpose

The purpose of this document is to present the description of the intended ePI Application Programming Interface (API) version 1.0.

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

2. Scope

The scope is defined as the new ePI API only.

This specification is based on the latest version of FHIR, available at <http://build.fhir.org/resourcelist.html>, built on top of R5 publication. See <http://hl7.org/fhir/directory.html> for a list of all FHIR versions.

3. Introduction

3.1. FHIR

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 the ePI API. 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 require changes to this specification while it is draft (but not once published and finalised).

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 the build server draft version (<http://build.fhir.org/resourcelist.html>).

3.2. Definitions

An API can be defined in various ways; the definitions below form a good start for this¹:

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 situation 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. Composition, MedicinalProductDefinition, Bundle, etc. (see 6.)

3.3. 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

3.4. 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.
- 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, versus 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.

3.5. Spelling

The resources and attributes in this specification are defined as per FHIR convention, which has standardised on US spellings. However, EMA uses British spellings, which have been used within this

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

135 specification in the descriptions and textual explanations. This generates a certain mismatch that is
136 however unavoidable.

137 **4. Specification**

138 The specification for the API is based on the RESTful style API. The same style of API was adopted for
139 the SPOR API 1.x and 2.0, PSUR API and for the Common Repository. This is for the sake of
140 consistency but also for its clarity, ease of use with minimal infrastructure and its clear separation
141 between resources and the operations that can be applied on those resources.

142 **4.1. Versioning**

143 **4.1.1. Versioning**

144 The ePI API is based on the FHIR specification. Both ePI API and FHIR will keep separate versioning
145 schemes and evolve at their own pace. Each version of the ePI API will be based on a particular version
146 number of FHIR, as recorded in this document.

147 This API will be up-versioned (for example, to become V1.01) if and when changes are necessary, in a
148 controlled and communicated manner. See <http://hl7.org/fhir/directory.html> for a list of version
149 numbers of FHIR release. FHIR servers communicate their supported version as part of their
150 conformance statement (see <http://hl7.org/fhir/capabilitystatement.html>).

151 **4.1.2. XML schemas versioning**

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

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

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

164 **4.1.3. Service versioning**

165 Each endpoint URL will be prefixed with /v{version}, where version is the service version number. The
166 service URL is case sensitive.

167 i.e. GET /v1/Bundle

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

170 **4.2. Authentication and authorisation**

171 All services require authentication, unless explicitly stated otherwise in the service definition.

4.3. 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 can use FHIR extensions, and these are documented within the individual specifications for those resources as used in this API.

4.4. HTTP methods

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

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

4.5. 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 <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:

Name	Code	Description	Comment
Read	200	OK	
Update			
Create	201	Created	
Create	202	Accepted	For an asynchronous operation, indicates initial basic success, with more work ongoing
Update			
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	400	Bad Request	Resource update failed basic validation or search parameters failed basic validation, or no id provided for update.
Update			
Create			
All	401	Not Authorized	Operation needs authorization and no authorization was attempted
All	403	Forbidden	Operation needs authorization and authorization failed
Read	404	Not Found	Unknown resource or unknown resource type (for Search, Update)
Search			
Update			

Name	Code	Description	Comment
Create			
Update	405	Method Not Allowed	Can't update a resource that didn't exist. Or not permitted to delete
Delete			
Update	422	Unprocessable Entity	The proposed resource (while basically valid) violated applicable FHIR profiles or server business rules.
Create			

Note that updates will never create a record that did not exist before.

Apart from targeting specific individual resources, Updates can also be achieved with transaction bundles (see [4.7](#)). These create or update multiple resources, and return a bundle of transaction results, each having an HTTP result code ([see 4.7.1.](#))

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>).

The server may also issue an OperationOutcome whenever the HTTP response code is a success. This would indicate that there are warnings or hints found during the business rules validation.

4.5.1. Asynchronous updates

Asynchronous operations are not considered in this ePI API specification. Details will be added in future when ePI is implemented into business.

4.6. 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 are used as working numbers for the day to day identification of ePI 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.

4.6.1. 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 do not 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/Bundle/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.)

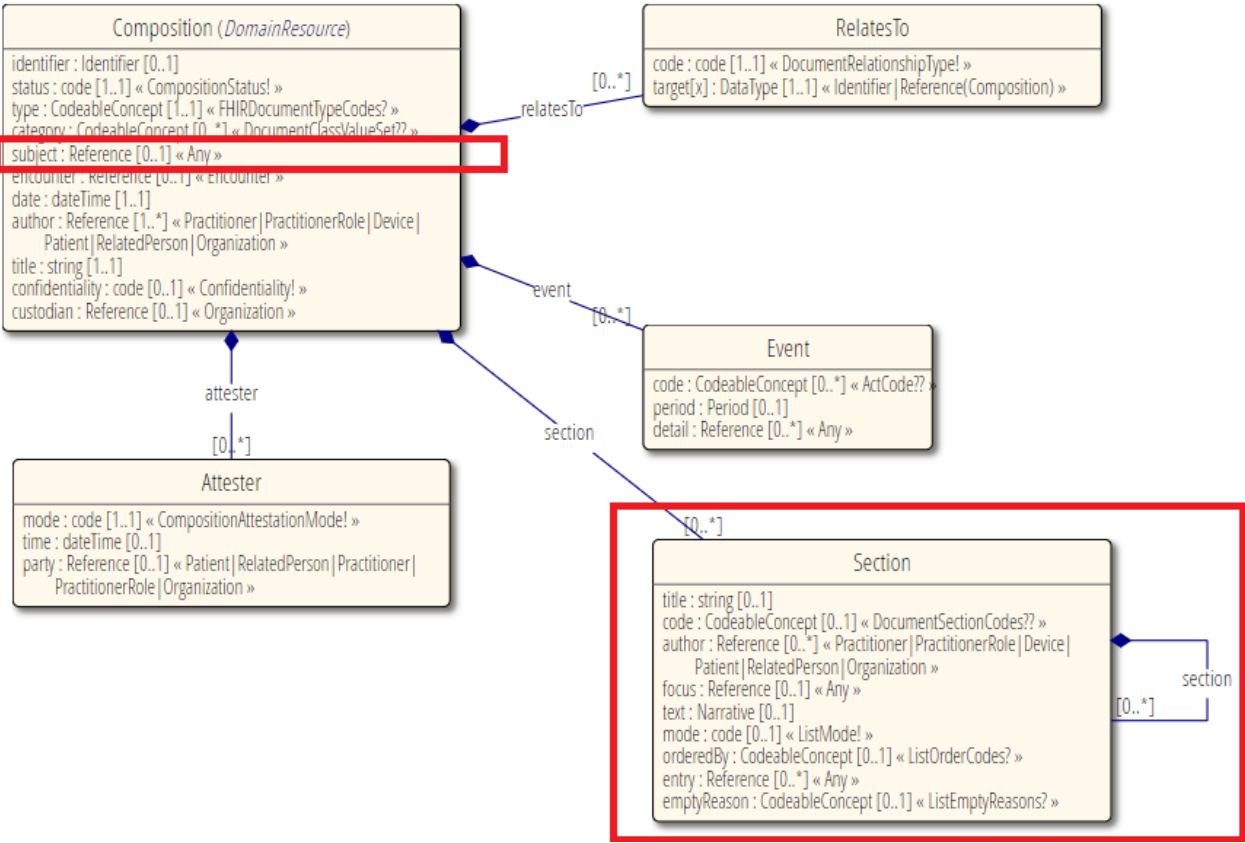
A software system that accesses the URL above will directly see the data for that single resource (or an error if the id is not recognised). This is a direct read access, with no searching or retrieval of other connected resources. Note that standard users will not normally see or interact with these ids and will never 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 ePI data itself – it is 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-struct>, but instead is covered here: <http://hl7.org/fhir/resource.html#tabs-struct>.

4.6.2. 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, Section is directly a part of the Composition resource, but the subject is not (it is a reference):



246

247 **Figure 1.** Example showing Section and subject.

248 This resource-oriented, or page-oriented view of data has implications, because the full relevant data
249 set is split over a series of resources, connected by references. Subject to business constraints of what
250 makes sense and is allowed, these can be treated either as separate resources, retrieved and updated
251 individually, or as a group of resources accessed together. In ePI, this is in use by combining
252 Compositions with other elements such as Lists or MedicinalProductDefinition to produce some higher
253 value outcome.

254 **4.7. Bundles**

255 The ePI API makes use of the FHIR bundle resource (see <http://www.hl7.org/fhir/bundle.html>).

256 Bundle is used in many APIs but has particular significance for ePI, as a document-oriented API, since
257 it is used as the basis for FHIR documents.

258 A bundle is a container resource that is used whenever a group of more than one resource is needed.

259 Each bundle has some basic header information, including its type (searchset, transaction, transaction-
260 response, batch or document), and a total number of “hits”, if it is a set of search results. It then
261 consists only of a repeating “entry” structure, each of which contains one resource (of any type), and
262 then possibly a request or result section, for use with transactions.

263 Some examples are:

264 • Documents

265 FHIR documents consist of a composition resource, for the headings, sections and text, and other
266 resources for the supporting structured information. For ePI, this information will be List resources,
267 containing references to MedicinalProductDefinitions (for the relevant products), and binary resources
268 for images. These are all wrapped in a bundle, of type “document”. All this represents one single
269 document, and these documents are the core entities of ePI (see 5. on representation of an ePI with
270 multiple document Bundles). Note that because bundles have other uses in FHIR, and this API can
271 refer to multiple documents at once, it makes use of Bundles of (document) Bundles.

272 • Search results

273 When receiving a bundle of 0 or more resources, of the type requested, and possibly other types that
274 are linked to that type (requested using “_include”). Also used when multiple resources are retrieved
275 using an operation such as \$everything.

276 • Transaction

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

283 • Transaction-response

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

286 • Batch

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

289 Bundle general schematic:

```
290 Bundle
291   type= transaction|transaction-response|searchset|batch|document
292   total=N (for searchset)
293   [entry
294     {resource}
295     [request (used in a transaction to give http commands)
296       method=POST|PUT|GET|DELETE
297     ]
298     [response (used in transaction response to give http results)
299       location={URL of resource, including id}
300     ]
301   ] *
```

302

303 4.7.1. Document bundles

304 Detail schematic of a document bundle, as used in this API:

```
305
306 Bundle
307   type=document
308   entry
309     Composition <!-- first resource must be a Composition -->
310     [contained
311       Binary <!-- images can be contained and referenced -->
312     ]
313     section
314       text "The HTML text of this section"
315   entry
316     List <!-- for this API, second resource is a List of product references -->
317     entry
318       reference MedicinalProductDefinition/{product-id}
319     (repeats)
```

320 4.7.2. Transaction bundles

321 Detail schematic of a transaction bundle, showing how linkages work:

```
322
323 Bundle
324   type=transaction
325   entry
326     {Parent Resource Type}
327     {attribute of child resource type}
328     reference=tempUuid (temp local id of child, gets replaced by server)
329   request
330     method=POST
331   entry
332     fullUrl=tempUuid (matching temporary local id)
333     {Child Resource Type}
334   request
335     method=POST
336
```

337 The result would be in this form:

```
338 Bundle
339   type=transaction-response
340   entry (one per created resource, same order as incoming transaction)
341     response
342       location={ParentResourceType/parentid} (URL says type and id of created
343 resource)
344     entry
345       response
346       location={ChildResourceType/childid}
```

347

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

349 **4.7.3. Bundle endpoints**

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

352 Bundles can also be retrieved from the other, resource-specific endpoints however. For example, a
353 search would return a set of resources in a searchset bundle.

354 **4.8. Searching**

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

357 e.g. GET /v1/Bundle?status=pending

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

361 It is also worth noting that searches will be performed against the latest version of the resource that
362 the caller is authorised to view. No match against historical information will be considered, but history
363 can be accessed via the "version" operations.

364 **4.9. Paging and sorting**

365 All FHIR results from a server are subject to paging. This is described here:

366 <http://hl7.org/fhir/http.html#paging>

367 This only affects results where there is more than one result (i.e. searches). It is possible to override
368 the default page size, by asking the server to supply more records per page using the "_count=N"
369 search parameter modifier.

370 e.g. GET /v1/Bundle?composition.type=100000155532&_count=100

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

376 e.g. GET /v1/Bundle?composition.type=100000155532&page=3

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

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

382 **4.10. Resources and representations**

383 For an API with a RESTful style, a resource is anything that can be identified and manipulated by a set
384 of HTTP verbs. Resources are defined by FHIR and referenced in the services in the rest of this
385 document (in particular, [see 7](#)).

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

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

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

397 If the representation requested is not supported by the server, then an appropriate error is returned by
398 the server to the client (see [4.5.](#))

399 Examples:

- 400 • Request for a resource representation in xml format: (may be omitted as default)

401 **Accept:** `application/fhir+xml`

- 402 • Request for a resource representation in JSON format:

403 **Accept:** `application/fhir+json`

404 **4.11. Encoding**

405 All resources are UTF-8 encoded, unless explicitly stated otherwise in the service definition.

406 **4.12. Request parameters and searches**

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

- 409 • **Path:** `/v1/[type]/{id}`

410 where the single parameter is the resource's FHIR id. [type] represents the name of a type of resource
411 e.g. Bundle. Note that resource names in FHIR are always case sensitive and in upper camel case.

• **Query string:**

`/v1/[type]?{param}={ [op]value[,value]} [&{param}={value}]`

e.g.: `/v1/[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 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.

4.12.1. 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.

4.12.2. 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>.

4.12.3. Chained searches

The ePI data is stored in FHIR as multiple resources, linked by FHIR references. A Bundle resource will contain a Composition and a List, which references products as MedicinalProductDefinitions, in a set of referenced PackagedProductDefinition resources. Compositions are never used on their own and API has no specific endpoint for them. Endpoints exist only for Bundle and for List.

Each endpoint can be queried. It is possible, and often necessary, 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>

449 An example in schematic form would be:

450 GET /v{version}/Bundle?{parent-param}={value}&{child-attribute-name}.{child-
451 param}={value}

452 In the above example, child-attribute-name is the name of the child resource when used as an
453 attribute in the parent resource. For example, the Bundle.composition is a reference to the
454 Composition within the document Bundle, and can be used to access its search properties e.g.
455 composition.title. Note that only search parameters can be accessed this way, not every attribute of
456 the resource.

457 A chained query could be:

458 GET /v{version}/Bundle?composition.title:contains=acmedrug

459 or

460 GET /v{version}/List?item:Bundle.composition.title:contains=acmedrug

461 Note that although chained queries can ask questions about data in a linked child resource, this is still
462 a query exclusively on the List resource and so will only return List resources and not the Bundle
463 resources that are queried, see also “_include” below.

464 **4.12.4. Including other resources in search results**

465 Every FHIR resource’s endpoint can be queried, as described elsewhere, and using the specific
466 parameters defined in this API. But each resource endpoint normally only fetches query results for that
467 particular resource type, not any others that may be linked to that data.

468 However, when searching it is possible for the results to include extra resources that are linked to the
469 parent. This uses the “_include” or “_revinclude” parameters. See

470 <http://www.hl7.org/fhir/search.html#include>

471 An example would be:

472 GET /List?item:Bundle.composition.title:contains=acmedrug&_include=List:item

473 Note that this includes the child resource by using the attribute name of it as used by the parent. It
474 does not use the name of the child resource type (which would be Bundle). The result will be a List
475 with references to Bundles, as well as the actual Bundles themselves.

476 **4.13. Metadata**

477 The metadata associated with resources is documented here:

478 <http://hl7.org/fhir/resource.html#metadata>.

479 Metadata includes the resource “versionId” and “lastUpdated” date. Both are available on every
480 resource.

481 “versionId” is the number of the FHIR history version. This is incremented with each save of a
482 resource. It cannot be queried directly, but is instead accessed by the “_history/{version-number}”
483 method.

484 “lastUpdated” is the server date of the last change to any data item in the resource and is also
485 therefore the date of the last “save”. It can be queried using a special query parameter called
486 “_lastUpdated”. This works in the same way as any other query parameter and is documented with
487 examples here: <http://www.hl7.org/fhir/search.html#all>.

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"?>
<Bundle 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>
  etc.
```

4.14. 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.

5. Resource structure

This API makes use of multiple resources to represent single business concepts. ePI is composed out of multiple documents (summary of product characteristics [SmPC], package leaflet, etc.). Every document is represented in the same way in FHIR.

5.1. Groups of resources

This diagram represents the structure of any document:

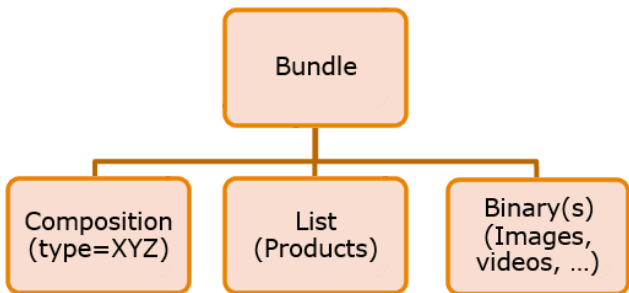


Figure 2. Structure of a document.

The type of the Composition identifies the type of document.

We can call these “document Bundles” to differentiate them from Bundles used for other purposes (envelopes and transactions) ([see also 4.7.1.](#)).

A document can be linked to other documents using a List of document Bundles.

The Composition resource represents the structure of the textual parts of the document. Nested Sections represent the structure of the document, shown bottom right of Figure 3.

The Section.text item, with type "Narrative", contains HTML and references to binaries (which are shown in Figure 2 as Binary resources). SPOR is used as master for controlled vocabularies, where OMS provides the link to the owner of the document (mapped in the author attribute) and RMS provides all of the terms of type Coding (for example document type, section type, etc.).

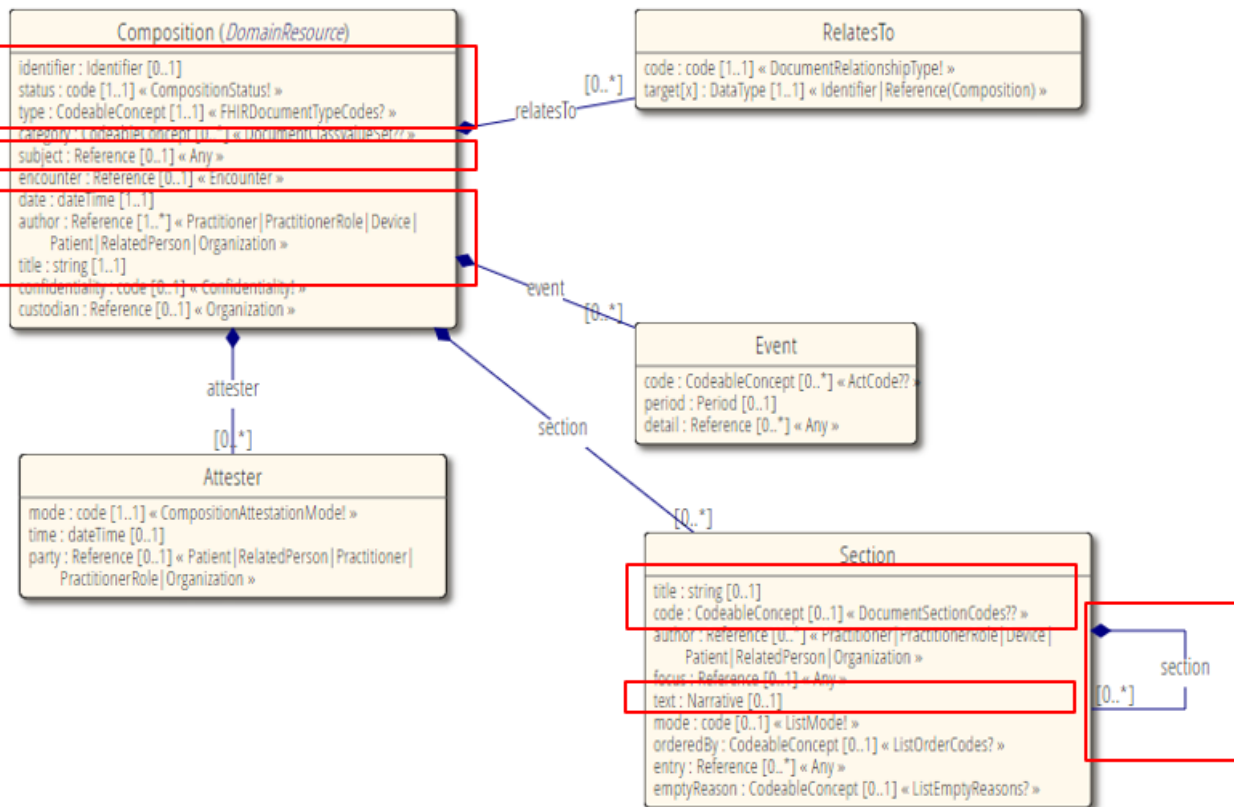


Figure 3. FHIR Composition resource.

An ePI set of documents is represented as shown in **Figure 4**.

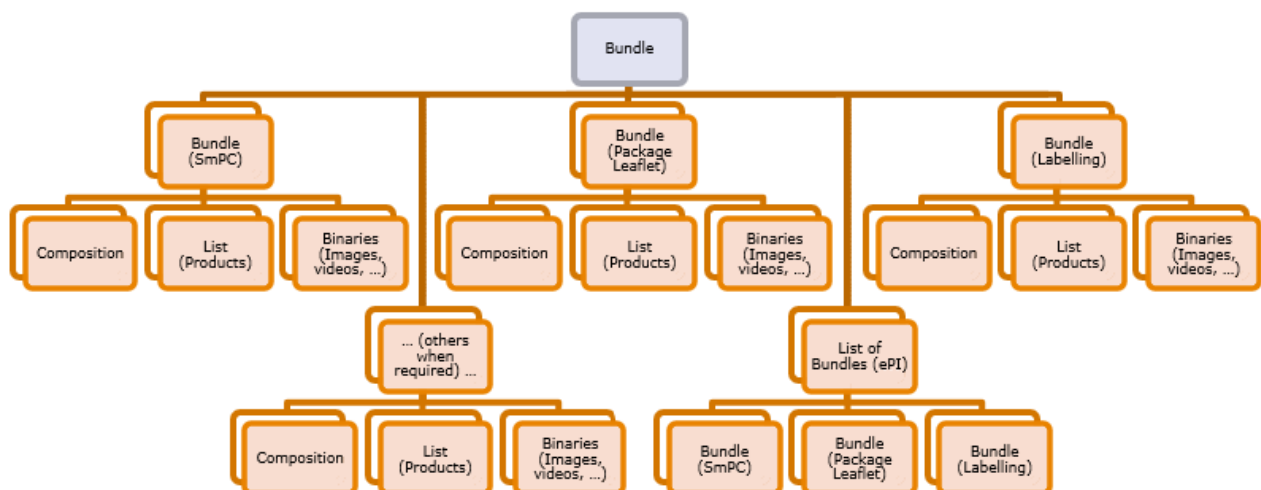


Figure 4. An ePI set of document Bundles, in an envelope Bundle.

A series of document Bundles is shown (SmPC, package leaflet etc.), each with its own components. Note that the grey "envelope" Bundle at the top is only used for the transport of the data to or from the server and it has no semantic meaning. It is not preserved after the data arrives.

The List resource at the bottom right ("List of Bundles (ePI)") acts as an index and is what groups the bundle documents together into the set for a single ePI.

5.2. Scenarios and lifecycle

This section describes scenarios of the ePI lifecycle and how they would be implemented with FHIR resources.

5.2.1. Scenario: creation with SmPC, package leaflet and 2 labelling entries

In **Figure 5.**, each orange bundle represents one whole document, whose internal structure is described earlier (see **Figure 2.** Structure of a document).

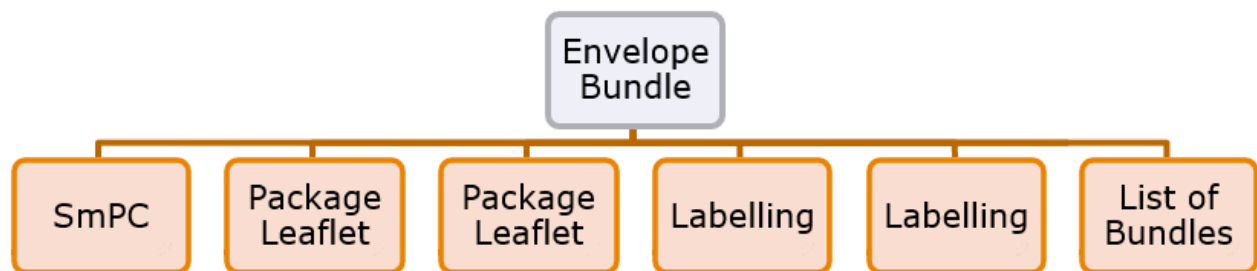


Figure 5. An ePI set of documents being created.

When creating a set of ePI documents on the server, an envelope Bundle is used to contain all the document Bundles in the set, as well as the indexing List of Bundles.

Points to note for this scenario:

- The "envelope" bundle disappears upon reception at the server, and plays no further part.
- SmPCs point to a List resource, of MedicinalProductDefinition resources. (This is different to the "index" List of Bundles).
- Package Leaflets also point to a List of MedicinalProductDefinitions.
- Labelling documents also can be linked to their own List of MedicinalProductDefinition
- The submitter has to submit the List of Bundles too, in all cases (shown at the far right, above). The receiving system can validate that this is present and correct.
 - The server will use profiles to check the content: ePI creation and ePI update.
- Submissions of a single document are allowed, but the List of Bundles must be maintained consistently.
- Different strengths of the same "product" could be in one single SmPC or multiple SmPCs. The standard allows that one ePI can have either one or multiple SmPCs.
- Business rules will enforce consistency of MedicinalProductDefinition linking in the different artefacts, according to the business process.

5.2.2. Scenario: creation of individual list entries

When it is not desirable to create all parts of the ePI at once, the previous scenario can be broken down into several steps:

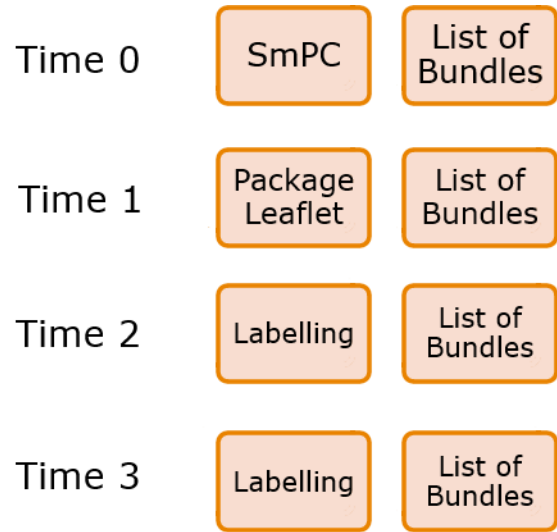


Figure 6. Creating an ePI document set in stages.

Here, each document is being created independently. One document per step is shown above, but each step can have one or more documents. The submitter has to submit the index List of Bundles together with every submission, and the pair will need to be in an envelope Bundle. The index List contains all the documents currently in the set plus any new ones - it always lists the full set of documents in every new step.

5.2.3. Scenario: adding a translation for the SmPC and labelling

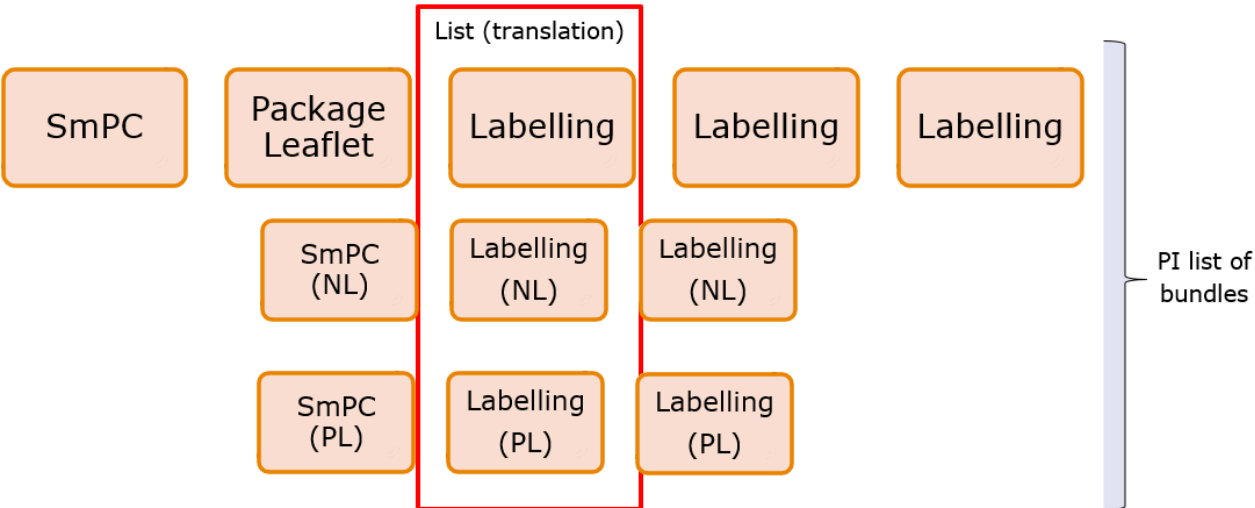


Figure 7. Adding document translations.

In this scenario, an ePI set of documents includes translations of the labelling. This requires an extra List resource, represented by the red outline. This is in addition to the document index List of Bundles, and is represented above by the grey grouping at the right-hand side. The diagram shows the "after" state of adding two sets of translations, NL and PL, possibly at different times.

The submitter of the translations to a document set is responsible for submitting the updated index List of Bundles, which acts to add the translations into the set index. (The scenario assumes that the ePI set exists on the server before the translations are added.) The submitter must also submit the "translation" List (in red in the diagram). The translation List is a sibling of the index List - in effect it is a translation index - and is external to the ePI set. It does not go in the main ePI index List itself.

Note that FHIR Compositions have a "relatesTo" element, that could be used to link a main document to its translations. However, this API avoids using that because of bi-directional issues when one document is not strictly the master, and also to model all relationships (ePI set index, translations index) in one homogeneous and more portable way: using Lists.

5.2.4. Scenario: updating a package leaflet

An ePI set is created as below, with a POST and with an envelope Bundle, and gets assigned ids by the server (shown on the two left hand document Bundles, but present on all). The mandatory List resource that links these together is shown as the grouping arrow on the right.

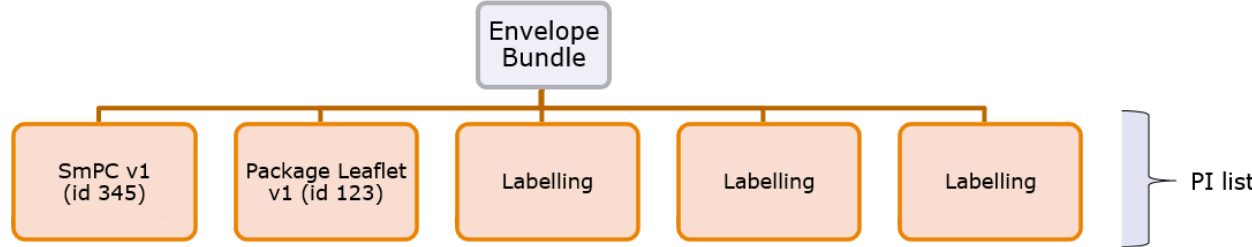


Figure 8. Initial ids are assigned by the server

In the above, a package leaflet is being sent to the server, as part of a Bundle of multiple documents. All of the above items can be sent together. The leaflet gets an id assigned, in this case: 123.

Note that doing this in one step implies use of a FHIR transaction, because the PI list must know the ids of the document Bundles, and they are not yet known when the List is being sent ([see 4.7.2.](#)).

To update a package leaflet (and the same applies to any other document type), a PUT is used. At this point the document bundles, including package leaflet 123 already exist on the server:



Figure 9. Update process

The PUT performs an update on item id 123, and replaces the content with what is sent. The package leaflet stays at id 123, but becomes version 2. Since item 123 is already in the PI list, that aspect is unchanged. No update to the index list is needed. All external links to id 123 will now point automatically to the new version of 123. This can happen again and again. There is no need to group changes in a Bundle or to submit any of the other documents in the original Bundle. The result is this:



Figure 10. After the update.

The SmPC can be updated similarly:



Figure 11. Update the SmPC.

With this result:



Figure 12. The SmPC has been updated.

5.2.5. Scenario: a new package leaflet and a labelling are added

A set of new documents can be added to an existing one. Here an extra package leaflet and labelling are being sent (POST new elements + PUT of the List of Bundles):

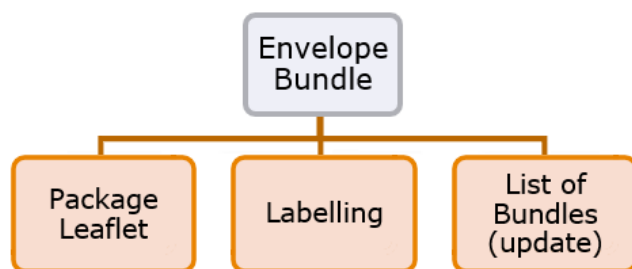


Figure 13. Adding new items to an existing ePI set.

Based on the set from the previous section, the result would be:

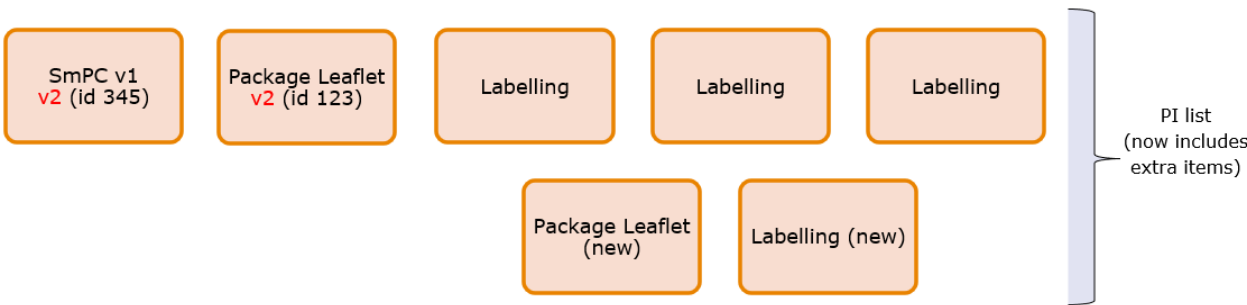


Figure 14. The ePI set with the additions.

5.2.6. Scenario: a labelling is deleted

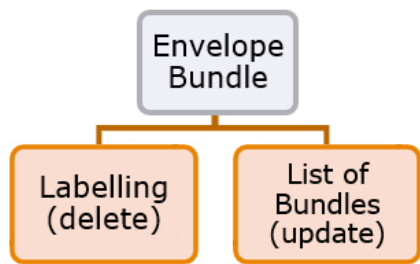


Figure 15. Delete an item.

To delete an item the DELETE http request is used to remove that resource, and the index list must also be adjusted to remove it from the set. These can both be done in one step using a Transaction Bundle as the envelope. The updated List references the old elements minus the deleted one.

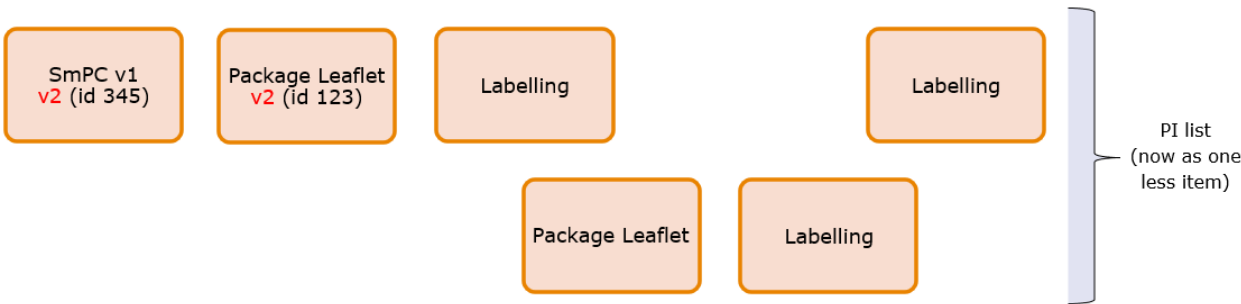


Figure 16. After the deletion.

5.2.7. Scenario: parallel distribution

The FHIR standard and resource structure for ePI offers great flexibility for implementation in varied business processes in the future. Use of ePI for parallel distributed medicines could possibly be accommodated as described here, however further consideration of the particular business process would be required. Parallel distribution involves different PI documentation for the same products. This can be accommodated by different ePI sets pointing at the same products. There is no other link between the two ePI sets.

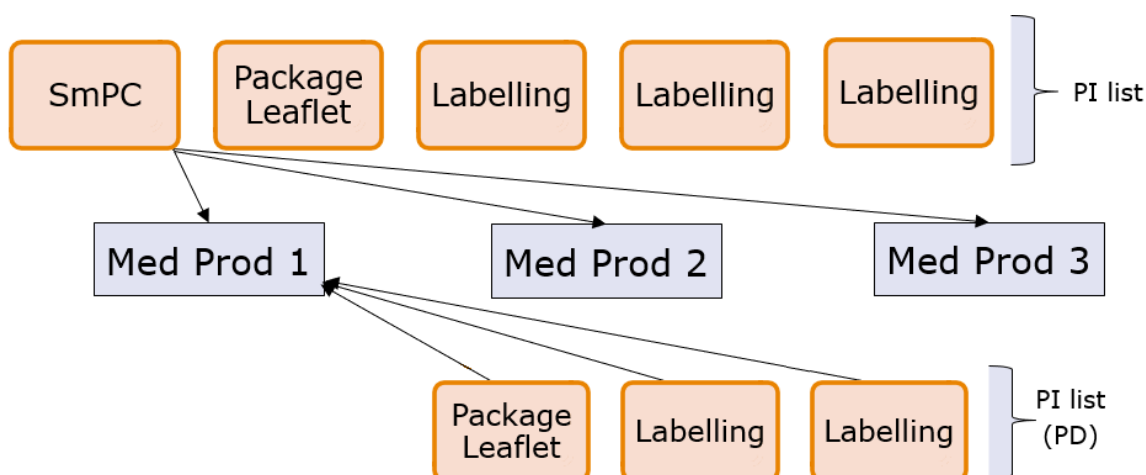


Figure 17. A Parallel distribution set points at the same products

6. REST services

6.1. Resource summary

A summary list of resources for this API is below (for full description, [see 7.](#)):

Resource	Description
Composition	Composition is the resource that represents the outline of a document, with metadata - such as the author - and repeating nested sections of HTML text. Sections can optionally refer to other resources for structured detail. Compositions are always contained in Bundles (of type "document"), and it is the Bundle that corresponds to the document as a whole.
Binary	Used for images, that can be embedded in the Composition and referenced from specific places in the HTML text.
List	A set of other resources (as resource references), with a specific coded type or purpose. This is used to represent: <ul style="list-style-type: none"> a set of products that are referenced in a document (a List of MedicinalProductDefinitions) a set of documents (a List of document Bundles)
MedicinalProductDefinition	Detailed definition of a medicinal product, typically for uses other than direct patient care (e.g. regulatory use). These appear as references in ePI, but not as full resources.
Bundle	A container for a collection of resources. Several uses in this API. ePI documents are Bundles (of a Composition and other resources). Bundles group together multiple resources - including other Bundles (document Bundles) - for search results, and when submitting multiple entities at once.

Other infrastructural resources such as CapabilityStatement and OperationOutcome may also be encountered and are covered in section 7Resources

649 **6.2. Service summary**

650 The service list is documented in a separate catalogue.

651 **7. Resources**

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

654 **7.1. Bundle**

655 <http://build.fhir.org/bundle.html> and [see 4.7.](#)

656 **7.2. List**

657 <http://build.fhir.org/list.html>

658 **Extension:**

Canonical URL	http://ema.europa.eu/fhir/extension/domain
Type	Identifier
Extends	List.subject

659 **Example:**

660 See next extension.

661 **Extension:**

Canonical URL	http://ema.europa.eu/fhir/extension/medicine-name
Type	Identifier
Extends	List.subject

662 **Example:**

```
<subject>
  <extension url="http://ema.europa.eu/fhir/extension/domain">
    <valueCoding>
      <system value="http://spor.ema.europa.eu/v1/100000000004" />
      <code value="100000000012" />
      <display value="H" />
    </valueCoding>
  </extension>
  <extension url="http://ema.europa.eu/fhir/extension/medicine-name">
    <valueCoding>
      <system value="http://spor.ema.europa.eu/v1/100000000005" />
      <code value="medicine-name-code" />
      <display value="Elocta" />
    </valueCoding>
  </extension>
</subject>
```


663
664 See next extension.

665 **Extension:**

Canonical URL	http://ema.europa.eu/fhir/extension/documentType
Type	Coding
Extends	List.entry.item

666 **Example:**

667 See next extension.

668 **Extension:**

Canonical URL	http://ema.europa.eu/fhir/extension/language
Type	Coding
Extends	List.entry.item

669 **Example:**

```
<item>
  <extension url="http://ema.europa.eu/fhir/extension/documentType">
    <valueCoding>
      <system value="http://spor.ema.europa.eu/v1/100000155531" />
      <code value="100000155532" />
      <display value="SmPC" />
    </valueCoding>
  </extension>
  <extension url="http://ema.europa.eu/fhir/extension/language">
    <valueCoding>
      <system value="http://spor.ema.europa.eu/v1/100000072057" />
      <code value="100000072178" />
      <display value="de" />
    </valueCoding>
  </extension>
  <reference value="Bundle/a156a584-2423-44a7-981b-c83c12e2ee7b" />
</item>
```

670 **7.3. Composition**

671 <http://build.fhir.org/composition.html>

672 **7.4. Binary**

673 <http://build.fhir.org/binary.html>

674 **7.5. MedicinalProductDefinition**

675 <http://build.fhir.org/medicinalproductdefinition.html>

7.6. OperationOutcome

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

<http://build.fhir.org/operationoutcome.html>

7.7. 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://build.fhir.org/capabilitystatement.html>

8. About this document

8.1. Definitions, acronyms, and abbreviations

Acronym/Abbreviation	Description
API	Application Programming Interface
ePI	Electronic Product Information
FHIR	Fast Healthcare Interoperability Resources
GUI	Graphical user interface
HTTP	Hypertext Transfer Protocol
IANA	Internet Assigned Numbers Authority
JSON	JavaScript Object Notation
MAH	Marketing Authorisation Holder
PSUR	Periodic Safety Update Report
SmPC	Summary of product characteristics
SPOR	Substances, Products, Organisations, Referentials
OMS	Organisations Management System
REST	Representational State Transfer
RMS	Referentials Management System
URL	Uniform Resource Locator
UTF-8	Unicode Transformation Format – 8-bit
UUID	Universal Unique Identifier
XML	Extensible Mark-up Language

8.2. Open issues

None.

8.3. Document approval

Date	Version	Submitted by	Approved by	Approve role

689 **8.4. Document history**

Version	Who	What
1.0	EMA	Creation for proof of concept and consultation in ePI set-up project
1.1	EMA	Extension domain moved from List.entry.item to List.subject. Extension medicine-name added. Typo fixed in Resource summary. Fictional product used in the examples instead of commercial brand.

690