

# Nictiz FHIR R4 Implementation Guide

## 1.0.4

---

## Inhoud

---

### Introduction

#### Overarching principles

- Functional definitions

  - Layering: zibs, nl-core profiles and information standard specific profiles

  - The zib layer

  - The nl-core layer

  - Relating FHIR profiles to its functional counterpart

  - Use of .mustSupport

- Profiling guidelines

- HTTP headers

  - Content Types and encodings

  - Multiple FHIR versions on the same server endpoint

- Use of coded concepts

  - Textual description of coded concepts

  - Code system URIs

  - Mapping of coded concepts

- Use of the reference data type

- Profile use and declaration

  - FHIR Packages

- Search

  - Search URLs and search parameters

  - Search results

- Usage of the .id, .identifier and .fullUrl elements in FHIR instances

  - .identifier versus .id

  - When is .identifier expected?

  - When is .id expected?

  - Logical ids, .fullUrls and references in Bundles

- Handling errors

- CapabilityStatements

- Including "secondary" resources when sending information

- Validation and terminology

  - SNOMED CT - Dutch edition

  - CodeSystems without concepts defined

- Missing information

- Resource.text or "the narrative"

  - "Clinically safe" narratives

  - Narrative status

### Release notes

# 1 Introduction

This implementation guide (IG) describes the requirements for using **HL7® FHIR®** in context of information standards that are maintained by Nictiz. It applies specifically to HL7® FHIR® version 4.0.1 (R4).

This IG is aimed at software vendors and developers that need to implement FHIR-based information standards maintained by Nictiz. Users of this guide are expected to be familiar with the [FHIR R4 specification \(https://hl7.org/fhir/R4\)](https://hl7.org/fhir/R4) and resource processing. Where relevant, links to the FHIR specification are provided. This implementation guide is not intended to be a tutorial on that subject. This IG uses the conformance verbs (SHALL, SHOULD, MAY) defined in the FHIR Conformance Rules. Requirements that use the verb 'SHOULD' are treated as 'comply or explain' during compliance testing ('qualification').

This IG describes overarching principles all Nictiz standards implemented in FHIR R4 SHALL conform to. However, information standard specific IGs or profiles can extend, specialize or overrule these principles if explicitly documented. These principles SHALL NOT break conformance to the FHIR Core specification.

## 2 Overarching principles

### 2.1 Functional definitions

---

Most, if not all, conformance resources are based on an underlying functional model. The functional model is the specification to which the conformance resources should adhere.

The basis for most other functional models is formed by the 'zibs' ('Zorginformatiebouwstenen') ([https://zibs.nl/wiki/HCIM\\_Mainpage](https://zibs.nl/wiki/HCIM_Mainpage)), in English also known as Clinical Information Models (CIMs), Health and Care Information Models (HCIMs) or Clinical Building Blocks (CBBs) - we will use the Dutch term 'zib' for all profiling work as it has become a recognizable term over the past years. The zibs were originally defined by the program 'Registratie aan de bron' (Data capture at the point of care). This program ended in April 2022 and maintenance and further development of zibs was transferred to the Nictiz' zib-centrum. Zibs provide a foundation of use case neutral building blocks from which use cases can be built. The formal definition of the zibs can be found on the [zibs wiki \(https://zibs.nl\)](https://zibs.nl)

Use cases or information standards use and refine zibs that are relevant to the situation. The formal specification for these information standards (a data set constrained with cardinalities and conformance applied, comparable to what FHIR calls a *logical model*) is linked to from the information standard specific IG.

#### 2.1.1 Layering: zibs, nl-core profiles and information standard specific profiles

The profiles and other conformance resources align to this layering of information standards. We recognize three levels of profiles:

**zib profiles**

profiles that represent the zib as faithfully as possible.

**nl-core profiles**

profiles derived from the zib profiles that might have been enriched by concepts from different use cases that apply the zib. See [#The nl-core layer](#) for more information.

**information standard specific profiles**

optional derived profiles from the nl-core profiles that further restrict or enhance these profiles for a specific use case.

#### 2.1.2 The zib layer

The goal of the zib layer is to create a faithful representation of each zib in FHIR. Because zibs are most often not

suitable for direct implementation, the profiles in the zib layer have `StructureDefinition.abstract` set to *true*.

### 2.1.3 The nl-core layer

The goal of the nl-core layer is to ensure that similar concepts defined at use case level are implemented in a uniform way at FHIR level. Whether the concept is used by a single or multiple use cases is irrelevant – adopting it at the intermediate nl-core level provides clear guidance for future use.

Concepts implemented at the nl-core level can be expected to be reused across use cases. Concepts that have little or no opportunity for reuse are expected to be defined at information standard specific level to avoid clutter at the nl-core level. Where possible, information standards aim to use nl-core profiles when exchanging FHIR resources.

### 2.1.4 Relating FHIR profiles to its functional counterpart

All profiles have a traceable relationship with their functional counterparts based on the element mapping mechanism in FHIR. This allows to:

1. define one or more references from a FHIR profile to an external URL where the functional definition can be found
2. define for each FHIR profile element to which concept *within* a functional definition it corresponds

The functional definition(s) underlying the profile can be resolved using the `StructureDefinition.mapping` (<http://hl7.org/fhir/R4/structuredefinition-definitions.html#StructureDefinition.mapping>) metadata field of a profile. To relate a FHIR profile field to a concept from the functional description, the concept ID defined in the `ElementDefinition.mapping` (<https://hl7.org/fhir/R4/elementdefinition-definitions.html#ElementDefinition.mapping>) can be used to look up the concept in the functional definition. It is possible that a single profile element refers concepts from multiple functional descriptions. Simplifier will show an overview of all the mappings for a FHIR profile on the dedicated 'Mappings' tab, together with a clickable URL to the functional definition. In addition, on the 'Overview' tab for each field the relevant mapping can be found.

### 2.1.5 Use of `.mustSupport`

The basic notion for any element or extension in a profile, most specifically when it has a mapping to a functional definition, is: provide what you have, and support what you can. At the generic profiling levels - zib and nl-core - we can not know what the exact context of use will be. We can therefore never be sure that a formal FHIR `mustSupport` (<http://hl7.org/fhir/r4/elementdefinition-definitions.html#ElementDefinition.mustSupport>) flag by any definition would hold in every imaginable use case. Therefore, `mustSupport` flags are absent in zib and nl-core profiles. The way to know the exact requirements for the use case at hand is thus not immediately clear from just looking at the generic profiles, but from reading the functional specification that should be part of any information standard.

An information standard may provide additional use case or transaction specific profiles, which constrain cardinality and apply `mustSupport` as applicable, for extra implementation guidance.

## 2.2 Profiling guidelines

---

All conformance resources created within Nictiz follow the conventions as outlined in the [FHIR Profiling Guidelines for FHIR R4](#).

## 2.3 HTTP headers

---

### 2.3.1 Content Types and encodings

The information standards use the [FHIR RESTful framework \(https://hl7.org/fhir/R4/http.html\)](https://hl7.org/fhir/R4/http.html). Of the formats defined (<https://hl7.org/fhir/R4/http.html#mime-type>), servers SHALL support both XML (MIME-type `application/fhir+xml`) and JSON (MIME-type `application/fhir+json`). Clients MAY use either format for the request and the response. This means that clients can indicate the desired response format using the optional `_format=[mimetype]` URL parameter, or the acceptable response format(s) using the Accept header. The URL parameter takes precedence over the header. If a client does not request a specific content-type, then it is server discretion to respond using XML or JSON.

FHIR uses UTF-8 for all request and response bodies. Since the [HTTP specification \(section 3.7.1\) \(https://www.w3.org/Protocols/rfc2616/rfc2616-sec3.html#sec3.7.1\)](https://www.w3.org/Protocols/rfc2616/rfc2616-sec3.html#sec3.7.1) defines a default character encoding of ISO-8859-1, requests and responses SHALL explicitly set the character encoding to UTF-8 using the charset parameter of the MIME-type in the Content-Type header. Requests MAY also specify this charset parameter in the Accept header and/or use the Accept-Charset header.

### 2.3.2 Multiple FHIR versions on the same server endpoint

Servers are allowed to support different FHIR versions on the same endpoint. For this reason, both servers and clients are encouraged to be explicit about the version used during any exchange. They also should be aware of the pitfalls of not being explicit; most notably, a client could encounter a response from the server in an unknown format when making informal assumptions about the FHIR version being used.

For clients, the following rules and guidelines apply:

- Clients SHALL use the appropriate FHIR version for the server endpoint. Clients may use any FHIR-based or other method for version discovery, including but not limited to:
  - Checking the `CapabilityStatement.fhirVersion` in the server `CapabilityStatement`: `GET [base]/metadata`.
  - Checking the server endpoint `[base]/$versions</base>`; note however that servers are not required to support this operation.
  - Using a (FHIR) service/endpoint registry.
- Clients SHOULD include the `fhirVersion` parameter in the Accept header of each request, for example: `Accept: application/fhir+json;fhirVersion=4.0`.
- Clients SHOULD include the `fhirVersion` parameter in the Content-Type header of each request that has this header, for example: `Content-Type: application/fhir+json;fhirVersion=4.0`.
- When both the Accept and Content-Type headers are used, clients SHALL use the `fhirVersion` parameter either in both or in none of these headers, and the value of the `fhirVersion` parameter SHALL match in both headers.

For servers, the following rules and guidelines apply:

- For servers that support a single FHIR version on their endpoint:
  - Servers MAY choose to ignore the `fhirVersion` in the Accept and/or Content-Type headers of incoming requests.
  - In case of a mismatch between the requested `fhirVersion` and the supported `fhirVersion(s)`, servers SHALL respond with HTTP status 400 Bad Request and SHOULD include an `OperationOutcome` with `.severity` set to *fatal* and `.code` set to *exception*, and SHOULD include a `.details` element that outlines the mismatch.
- For servers that support multiple FHIR versions on their endpoint:
  - Servers SHALL process requests in accordance with a matching `fhirVersion` parameter in the Accept and Content-Type headers of incoming requests. This means that they SHALL format data according to the profiles specified for the matching FHIR version.
  -

Servers SHALL support a default version in absence of the `fhirVersion` parameter in the Accept and Content-Type headers.

Note: When a server previously supported FHIR STU3 on an endpoint, it makes sense to set STU3 as the default version to remain compatible with existing clients. Newer R4 implementations of clients are expected to use the `fhirVersion` parameter, whereas this was not the case for STU3 implementations.

- In case of a mismatch between the requested `fhirVersion` and the supported `fhirVersion(s)`, servers SHALL respond with HTTP status 400 Bad Request and SHOULD include an `OperationOutcome` with `.severity` set to *fatal* and `.code` set to *exception*, and SHOULD include a `.details` element that outlines the mismatch.

## 2.4 Use of coded concepts

---

### 2.4.1 Textual description of coded concepts

Coded information is important for interoperability between systems. The zibs prescribe which ValueSets and CodeSystems to use for all its concepts.

However, it is likely that unknown codes present themselves in the communication, for example because the sending system has updated its medication codes to a newer version than the receiving system, or because an older record item is coded in a way that now has been deprecated. It is therefore vital that, in addition to the code itself, the sender includes the semantics of the code. Therefore, systems SHOULD include `(CodeableConcept).coding.display` (<https://hl7.org/fhir/R4/datatypes.html#coding>) and/or `CodeableConcept.text` (<https://hl7.org/fhir/r4/datatypes.html#codeableconcept>) for coded data in FHIR instances. These fields may only be absent in rare circumstances, for example when dealing with historical data, or in data acquired without such info from a third party. If multiple *codings* are available, at least one `.coding.display` SHALL be available, which SHALL be the value most relevant to the use case.

If `(CodeableConcept).coding.display` cannot be retrieved from the relevant code system (<https://www.hl7.org/fhir/r4/datatypes.html#Coding>), it may be retrieved from the ValueSet that is bound to the relevant element. Any other value for `.coding.display` will trigger a warning during validation. The semantic validity of the display value will then have to be assessed.

The spec on `CodeableConcept` (<https://hl7.org/fhir/R4/datatypes.html#CodeableConcept>) offers guidance on how the presence of `CodeableConcept.text` can be handled. Usually, the value of `.text` will be preferred above `.coding.display`, unless one of the *codings* is flagged with `.userSelected = true`. A `.coding` equivalent to *other* (often `.code OTH` in `.system http://terminology.hl7.org/CodeSystem/v3-NullFlavor`) leads to `.text` containing the actual meaning of the concept. For example: given an element for eye color with a bound ValueSet with concepts *blue*, *brown* and *OTH*, if `.coding.code` is filled with *OTH*, `.text` is to be filled with *green* (or any other color that is not represented in the ValueSet). Systems SHALL present the value of `.text` to its users.

### 2.4.2 Code system URIs

A coded value in FHIR is communicated as a combination of a code (in `.code`) and the canonical URI for the code system from which it was drawn (in `.system`). Implementers SHALL use the canonical URI that can be found by inspecting the ValueSet that is bound to a coded element in the appropriate FHIR profile<sup>[1]</sup>. Please note that a code system URI *can* align with its OID (as a `urn:oid:[oid]` URI), but that this is not always the case. For example, SNOMED CT is referenced by URL rather than by `urn:oid:[oid]`<sup>[2]</sup>.

## 2.4.3 Mapping of coded concepts

Terminology from zibs or UCSDs (use case specific data sets) needs to be faithfully applied to the FHIR profiles, but this is not directly possible when the base FHIR resource defines required terminology that bears no direct relationship with the zibs or UCSD. These cases include, but are not limited to, the FHIR resource elements of data type *code* (for example `Patient.gender`, `status` elements, structural type elements). This data type always describes FHIR specific terminology, which usually doesn't align with the zibs/UCSD terminology.

In these cases, a mapping will be provided to explain the relationship between the zib/UCSD terminology and the FHIR terminology. This is normally done through a FHIR [ConceptMap](https://hl7.org/fhir/R4/conceptmap.html) (<https://hl7.org/fhir/R4/conceptmap.html>), which can translate between the terminology used in the zib/UCSD, and the terminology that FHIR instances need to adhere to. This ConceptMap will be linked to the existing ValueSet using the [permitted-value-conceptmap](https://www.hl7.org/fhir/extension-11179-permitted-value-conceptmap.html) extension (<https://www.hl7.org/fhir/extension-11179-permitted-value-conceptmap.html>). When the ConceptMap equivalence is not *equal* or *equivalent*, the terminology from the zib/UCSD SHALL be communicated using the extension [ext-CodeSpecification](https://simplifier.net/resolve?target=simplifier&canonical=http://nictiz.nl/fhir/StructureDefinition/ext-CodeSpecification) (<https://simplifier.net/resolve?target=simplifier&canonical=http://nictiz.nl/fhir/StructureDefinition/ext-CodeSpecification>) on the same element. It is recommended to use this extension in other equivalence situations as well. This allows unambiguous interpretation on the receiver end.

Example snippet from an instance of [AllergyIntolerance](https://simplifier.net/resolve?target=simplifier&canonical=http://nictiz.nl/fhir/StructureDefinition/nl-core-AllergyIntolerance) (<https://simplifier.net/resolve?target=simplifier&canonical=http://nictiz.nl/fhir/StructureDefinition/nl-core-AllergyIntolerance>) where in `.criticality` both the FHIR terminology *high* and the SNOMED CT *24484000* terminology are communicated. The FHIR terminology is obtained from the zib terminology using [ConceptMap MateVanKritiekZijnCodelijst\\_to\\_AllergyIntoleranceCriticality](https://simplifier.net/resolve?target=simplifier&canonical=http://nictiz.nl/fhir/ConceptMap/MateVanKritiekZijnCodelijst-to-AllergyIntoleranceCriticality) (<https://simplifier.net/resolve?target=simplifier&canonical=http://nictiz.nl/fhir/ConceptMap/MateVanKritiekZijnCodelijst-to-AllergyIntoleranceCriticality>):

```
<criticality value="high">
  <extension url="http://nictiz.nl/fhir/StructureDefinition/ext-CodeSpecification">
    <valueCodeableConcept>
      <coding>
        <system value="http://snomed.info/sct"/>
        <code value="24484000"/>
        <display value="Severe"/>
      </coding>
    </valueCodeableConcept>
  </extension>
</criticality>
```

## 2.5 Use of the reference data type

A key feature of FHIR is the ability of resources to reference each other. This is done using the [Reference](https://hl7.org/fhir/R4/references.html#Reference) data type (<https://hl7.org/fhir/R4/references.html#Reference>). This data type supports two modes of referencing:

### **Literal references** (<https://hl7.org/fhir/R4/references.html#literal>), using the `.reference` element

A relative or absolute REST endpoint containing the `.id` of the referenced resource. In a Bundle context, this may also be a reference to a `Bundle.entry.fullUrl`.

### **Logical references** (<https://hl7.org/fhir/R4/references.html#logical>), using the `.identifier` element

Meaning a match on the business identifier (`.identifier`) for the referenced resource, without specifying where to find the referenced resource. Note: confusingly, although this reference type is called "logical reference", it does *not* act on the "logical identifier" (`.id`) of resources.

Profiles usually do not constrain which resources may be targeted from a given element. For example, `Condition.subject` may point to a Patient or Group resource. The [nl-core-Problem](https://simplifier.net/resolve?target=simplifier&canonical=http://nictiz.nl/fhir/StructureDefinition/nl-core-Problem) (<https://simplifier.net/resolve?target=simplifier&canonical=http://nictiz.nl/fhir/StructureDefinition/nl-core-Problem>) profile adds `nl-core-Patient` as a third option, to adhere both to [open world modelling](#) and a need to provide useful guidance on how this reference should be handled. However, use case specific profiles may constrain the target profiles.

The basic requirements for using references in this context are:

- Either a literal or logical reference SHALL be specified.
  - Literal references are preferred over logical references where possible.
- Literal references SHALL be resolvable.
  - External references SHALL be regarded in the same context as the resource itself. If the current security context is insufficient for retrieving the reference, then additional negotiation for appropriate privileges may be required.
  - Relative references are preferred over absolute references.
- For a receiver it is not always possible or feasible to actually resolve a reference. Therefore, a short description of the target resource SHOULD be included using the `.display` element.  
Please note: in rare circumstances, it is undesirable to provide a receiver with detailed information until it is actually able to resolve the reference, e.g. due to privacy considerations. In this case, it is expected that the `.display` contains an explicit remark that this information is masked.
- The resource type of the target resource SHOULD be included using the `.type` element.

In some circumstances, the constraints on the target resource type are too strict for the use case and the appropriate resource can't be referenced. In such cases, the proper reference type is added using an extension within the element that is used to refer the other resource. To aid receiving systems that cannot handle the extension, the sending system SHOULD give as much information as possible in the main element, for example by repeating the `.display` used in the extension in the main element.

## 2.6 Profile use and declaration

---

The implementation guides for the various information standards list the package version and profiles to use for the FHIR resources. A profile is a statement about the rules that this resource was created against. In the interest of interoperability it is important that, when using said information standard:

- Each resource SHALL be a valid instance of the applicable profile. Note that this applies not only to profiles listed directly in the IG, but also to profiles referred from/used by listed profiles.
- Each resource SHALL include the canonical URL of this profile in the `.meta.profile` element. If the applicable profile is an information standard specific profile derived from an nl-core profile, the canonical URL of the latter SHALL be included in an additional `.profile` element. The canonical of compatible base or derived profiles MAY be included as well.
- A receiving system MAY use the stated value of `.meta.profile` for validation, documentation or other purposes.
- A receiving system MAY use other profiles than stated in `.meta.profile`, e.g. when it has derived more constrained profiles based on the stated `.meta.profile`. Reasons for additional validation may include protecting the integrity of the receiving system or determination of internal process flow.
- The use of a version (appended to the canonical URL, separated by a vertical bar `|`) is strongly discouraged, because declaring compatibility this way is incompatible with any SemVer Grammar (e.g. it is not possible to declare a `|1.x` compatibility in a *canonical*).

### 2.6.1 FHIR Packages

The information standards maintained by Nictiz adopt the FHIR Packaging mechanism to support consistent versioning of profiles and related conformance resources such as OperationDefinitions. FHIR Packaging is based on the NPM Packaging mechanism and offers developers a convenient way to include the conformance resources in their favorite IDE. The relevant package version is indicated and linked in the information standard specific IG.

Please note that every effort has been made to ensure that the examples contained in the packages are correct and useful, but they are not a normative part of any information standard.

For more background information:

- The HL7 base specification for FHIR packages (<https://confluence.hl7.org/display/FHIR/NPM+Package+Specification>)
- What are FHIR Packages? (<https://registry.fhir.org/learn>)
- What is the problem that packaging could address (<https://blog.fire.ly/2017/11/28/versioning-and-canonical-urls/>)

**Note:** It is **not** required to implement FHIR-based information standards using the packaging mechanism. It is still possible to download all or selected resources from Simplifier on as-needed basis. You are however encouraged to invest in dealing with packages.

## 2.7 Search

---

### 2.7.1 Search URLs and search parameters

Each query use case of the various information standards lists the search parameters to use for a client to query the relevant information on a server. Usually, this is done in the form of (a number of) search URL(s), although individual use cases may use some other form if necessary (for example, listing optional parameters). Regarding the search parameters, the following guidelines apply:

- Unless explicitly marked as optional, the listed search parameters are considered the minimum for a request, meaning that:
  - All stated parameters and modifiers SHALL be supported both by clients and servers.
  - All stated parameters and modifiers SHALL be used for searching as specified in the information standard.
- A client MAY use additional search parameters in the search (such as date-range) not defined in the information standard. These may serve to further constrain the result set.
- A server MAY ignore and MAY support additional search parameters. A server SHALL report the applied parameters in the self link (<https://www.hl7.org/fhir/R4/search.html#conformance>) of the response and SHOULD provide additional information about the search process using an OperationOutcome in the search result Bundle (as an entry with `.search.mode` set to *outcome*). A client MAY use the self link and/or OperationOutcome to inform the user, independently apply these filters to the result set, or take some other action.
- A server SHALL reject any search request that contains parameters suffixed by a modifier that the server does **not** support for that parameter. For example, if the server supports the name search parameter, but not the `:exact` modifier on the name, it should reject a search with the parameter `name:exact=Bill`, using an HTTP 400 error with an OperationOutcome with a clear error message.
- A server SHALL NOT return information that is out of scope for the defined use case or authorization, even if the request by the client is wider than the use case.
- Repeating search parameters constitute a logical AND. Repeating search parameter values constitute a logical OR. Where applicable, this will be described further in the implementation guide.
- The order of the search parameters is always irrelevant.

General guidance on the use of search parameters can be found in the FHIR specification (<https://www.hl7.org/fhir/R4/search.html>).

#### 2.7.1.1 Search on date, number or quantity

When a search parameter has a type of date, number or quantity, the request uses prefixes (<https://www.hl7.org/fhir/R4/search.html#prefix>) to control the search behavior. Unless explicitly stated otherwise, the minimum set of supported prefixes for these types is `eq`, `gt`, `lt`, `ge` and `le`. For example:

```
GET [base]/Observation?date=ge2019-01-01&date=le2020-01-01
```

searches on all Observation resources that have an `.effective[x]` between January 1, 2019 and January 1, 2020.



## 2.7.2 Search results

### 2.7.2.1 Including referenced resources

It is possible that servers do not support read interactions, but all references from search results need to be resolvable (see [#Use of the reference data type](#)). Therefore servers MAY choose to include the referenced resources in the `searchset` Bundle, even if the client didn't use any `_include` parameters. These resources SHALL have `Bundle.entry.search.mode` set to `include`. Servers SHOULD NOT use contained resources for this purpose.

### 2.7.2.2 Paging

Servers MAY use [paging](https://www.hl7.org/fhir/R4/http.html#paging) (<https://www.hl7.org/fhir/R4/http.html#paging>) for performance reasons. Clients SHALL support this mechanism to prevent loss of information.

## 2.8 Usage of the `.id`, `.identifier` and `.fullUrl` elements in FHIR instances

---

### 2.8.1 `.identifier` versus `.id`

FHIR recognizes two fields that are used as identifier for instances: `.identifier` and `.id`. Although these are both identifiers, they are unrelated and serve a completely different purpose:

- `.identifier` is a business identifier, which usually has a meaning *outside* of the server. Examples are a registration number of a healthcare provider, a BSN or social security number for citizens, ISBNs for books, etc. Any instance may have multiple kinds of identifiers.
- `.id` is the logical identifier, or technical identifier, akin to the `id` field in a database. It is used as a unique handle for an instance of a given type on a particular server, and is needed to construct the URL to the instance. As such, it is used for referring between resources. The `.id` has no further meaning outside of the server.

### 2.8.2 When is `.identifier` expected?

Source systems SHALL be responsible for assigning a business identifier to every identifiable information object. Source systems SHALL have a methodology in place to guarantee global uniqueness. This business identifier is assigned to the `.identifier` element of FHIR instances when sending the resource, using a `.system` (e.g. a URL or `OID` that is under control of the sending organization). Because in HL7v3 (CDA) an identifier can only be composed using OIDs, the `.system` should preferably be an `OID` to accommodate compatibility in transformations from FHIR. This is especially true for resources that are used in information standards that have an HL7v3 (CDA) component, such as the BgZ, GP data and Medication process.

Secondary systems, that are not the source of the resource, SHALL be responsible for maintaining the business identifier assigned by the source system as they obtained it. The presence of the `.identifier` element helps receiving systems with re-identification and deduplication of resources.

### 2.8.3 When is `.id` expected?

As stated above, the logical `id` is meant to uniquely identify instances on a particular server; it is a vital component when using FHIR within a RESTful context. So as a rule of thumb, the `.id` element should always be present when dealing with instances that have a logical `id`, thus with instances on a server. This means:

- When a client reads, updates or otherwise addresses an existing resource on a server, the `.id` element SHALL

be populated in the request and response, and it SHALL match the id in the request URL.

- When a client sends a *new* instance to a server using a create operation, the `.id` element is *not* expected to be present -- these new instances don't exist yet on the server, so a logical id has no meaning. However, the server SHALL assign a logical id and populate the `.id` element in the response.

*Note:* It is not strictly prohibited to populate the `.id` field when sending a new instance, but the server SHOULD ignore it.

- When a client performs a search on a server, all instances in the returned searchset Bundle SHOULD have the `.id` field populated.

*Note:* A server SHOULD try its best to create stable ids for the resources it serves. Omitting `.id` elements is strongly discouraged as it breaks the assumptions about RESTful behavior. However, a server that omits `.ids` is still considered conformant when the conditions below are met:

1. The server does not natively support logical ids, for example when it is a stateless middleware server that gets its input from a non-FHIR backend.
2. *And* the use case doesn't require read/update/delete/patch support for any of the returned or referenced resources, as stated by the CapabilityStatement for the information standard.
3. *And* the server includes all referenced resources in searchset Bundles (regardless of whether the client asked to do so). These resources refer to each other based on the `fullUrl` mechanism as described below.

## 2.8.4 Logical ids, `.fullUrl`s and references in Bundles

There are several FHIR operations, like searching or batch create operations, where multiple instances are placed together in a Bundle. These instances will usually contain references to each other. These internal references cannot be resolved simply by inspecting the `.id` element of every instance; it is not guaranteed that all instances have a logical id. Instead, Bundles allow an `.entry.fullUrl` element for each instance which may be used for references. This is an additional mechanism to the logical id, not a replacement:

- The guidelines above for populating the `.id` field still apply.
- If `.id` is present, `.fullUrl` SHALL correspond with it (see below).

`.fullUrl`s may be RESTful URLs, UUIDs or even OIDs, based on the situation (which may be mixed within the same Bundle):

### Instances with a logical id

If an instance can be accessed on the server using RESTful operations, the `.id` of that instance in the Bundle will be populated. The corresponding `.fullUrl` in this case SHALL be the absolute URL to the instance on the server.

Instances within the Bundle may use relative references to each other, like they are on the same server. The FHIR machinery specifies how to find the matching instance based on the `.fullUrl`.

### Instances without a logical id

When instances are to be created on a server or when the server doesn't support reading individual instances, they don't have a logical id. When these id-less instances need to be referenced from within a Bundle, there are two alternatives:

- UUIDs can be used as single-use ids that will change each time the Bundle is generated. The `.fullUrl` for the instance will be the UUID prefixed with `urn:uuid:`. <sup>[3]</sup>
- OIDs can be used if the instance has an OID-based *business* identifier (i.e. the `.identifier` field), like the UZI number of a healthcare provider. The `.fullUrl` will be the OID prefixed with `urn:oid:`. <sup>[3][4]</sup>

Instances within the Bundle should use the prefixed version of the UUID/OID for referencing.

### 2.8.4.1 Example

Consider a client that wants to send a new Observation instance to a server and simultaneously link it to an existing Task instance. This can be done using a transaction Bundle:

```
<Bundle xmlns="http://hl7.org/fhir">
  <!-- Transaction Bundle that will simultaneously create a new instance and update an existing one -->
  <type value="transaction"/>
</Bundle>
```

```

<!-- A new instance to create -->
<entry>
  <!-- The new instance doesn't have a Logical id yet, so a temporary UUID is used for references within the
Bundle -->
  <fullUrl value="urn:uuid:0e855422-b8ef-4247-9443-f3747e78747e"/>
  <resource>
    <Observation>
      <!-- The id field is absent, because the instance doesn't exist yet on the server -->
      ...
    </Observation>
  </resource>
</entry>
<request>
  <method value="POST"/>
  <url value="Observation"/>
</request>

<!-- An existing instance to update with a reference to the new instance -->
<entry>
  <!-- RESTful URL of existing Task instance -->
  <fullUrl value="http://example-xis.com/Task/1234"/>
  <resource>
    <Task>
      <!-- Logical id of the instance, matches the fullUrl -->
      <id value="1234"/>
      ...
    </Task>
  </resource>
</entry>

```

The server response should then look like:

```

<Bundle xmlns="http://hl7.org/fhir">
  <type value="transaction-response"/>

  <entry>
    <!-- The newly created instance gets the Logical id "5678" from the server -->
    <fullUrl value="http://example-xis.com/Observation/5678"/>
    <resource>
      <Observation>
        <!-- The id matches the fullUrl -->
        <id value="5678"/>
        ...
      </Observation>
    </resource>
  </entry>
<response>
  ...
</request>

<entry>
  <fullUrl value="http://example-xis.com/Task/1234"/>
  <resource>
    <Task>
      <id value="1234"/>
      ...
      <output>
        ...
        <valueReference>
          <!-- Reference to the created instance may be relative -->
          <reference value="Observation/5678"/>
        </valueReference>
      </output>
    </Task>
  </resource>
</entry>

```

## 2.9 Handling errors

Errors in FHIR interactions are usually communicated using the combination of an appropriate HTTP status code in the 4xx-5xx range, and an OperationOutcome (<https://hl7.org/fhir/R4/operationoutcome.html>) resource providing more details regarding the error. The OperationOutcome resource contains mandatory elements to indicate the overall type and severity of the error (respectively `.code` and `.severity`) and MAY express the location (using `.location` or `.expression`) and a free-form or coded description of the error (using the `.diagnostics` or `.details` element). Although these latter elements are not required, their usage is strongly encouraged.

For the most common errors, the expected handling is listed in the table below. This should be interpreted as a

guideline, as the nature of an error is not always unambiguous and various FHIR reference implementations will make different choices. The FHIR specification provides more detail on error handling in general (<https://www.hl7.org/fhir/R4/http.html>), access denied responses (<https://www.hl7.org/fhir/R4/security.html#AccessDenied>) and for search operations specifically (<https://www.hl7.org/fhir/R4/search.html#errors>).

In general, most FHIR-related errors (in addition to normal HTTP errors related to security, header and content type negotiation issues) will result in one of these HTTP status codes:

- 400 Bad Request - resource could not be parsed, search could not be processed or basic FHIR validation rules failed
- 401 Unauthorized - authorization is required for the interaction that was attempted (usually emitted by the authorization server)
- 403 Forbidden - the server was unable to execute the search, due to an authorization failure (usually emitted by the resource server)
- 404 Not Found - resource type not supported, or not a FHIR endpoint
- 422 Unprocessable Entity - the proposed resource violated applicable FHIR profiles or server business rules

Interaction	Error	Expected action	Example
All	Not authorized, absent or expired access token	<ul style="list-style-type: none"> <li>HTTP 401 Unauthorized</li> <li>OperationOutcome with .code set to <i>security</i>.</li> </ul>	<a href="#">Example</a>
	Invalid authorization, scope of the access token is not sufficient for the request	<ul style="list-style-type: none"> <li>HTTP 403 Forbidden</li> <li>OperationOutcome with .code set to <i>security</i>.</li> </ul>	<a href="#">Example</a>
	Resource is not supported	<ul style="list-style-type: none"> <li>HTTP 404 Not Found</li> <li>OperationOutcome with .code set to <i>not-supported</i>.</li> </ul>	<a href="#">Example</a>
search	No match for the given search parameters	<i>This is not considered an error, but a normal search without results.</i>	
	Unknown search parameter	<i>This is not considered an error. See the <a href="#">guidance in the section on searching</a>.</i>	
	Syntactically incorrect parameter	<ul style="list-style-type: none"> <li>HTTP 400 Bad Request</li> <li>OperationOutcome. There are various issue type codes that could be used, depending on the nature of the error</li> </ul>	<a href="#">Example</a>
	Known but unsupported value. This situation applies to non-supported profiles when the Resource endpoint itself is supported.	<ul style="list-style-type: none"> <li>HTTP 200 OK</li> <li>Empty Bundle of .type <i>searchset</i>.</li> <li>OperationOutcome in Bundle as an entry with Bundle.entry.search.mode set to <i>outcome</i>.</li> <li>OperationOutcome with .code set to <i>not-found</i> and .severity not set to <i>fatal</i> or <i>error</i>.</li> </ul>	<a href="#">Example</a>
read	Read request with unknown id	<ul style="list-style-type: none"> <li>HTTP 404 Not Found</li> <li>OperationOutcome with .code set to <i>not-found</i>.</li> </ul>	<a href="#">Example</a>
update	No or incorrect Resource id	<ul style="list-style-type: none"> <li>HTTP 400 Bad Request</li> <li>OperationOutcome with .code set to <i>invalid</i>.</li> </ul>	<a href="#">Example</a>
create/ update	Resource syntax or data is incorrect, invalid or unsupported	<ul style="list-style-type: none"> <li>HTTP 400 Bad Request or HTTP 422 Unprocessable Entity</li> <li>OperationOutcome with .code set to <i>invalid</i> or preferably a more specific child code.</li> </ul>	<a href="#">Example</a>

## 2.10 CapabilityStatements

Information standards may have a [CapabilityStatement](https://www.hl7.org/fhir/stu3/capabilitystatement.html) (<https://www.hl7.org/fhir/stu3/capabilitystatement.html>) that is informative in nature and does not represent the minimum or maximum set of capabilities the client or server should support. Nictiz will strive to provide CapabilityStatements that are as complete as possible, however for the exact set of capabilities the implementation guide of the corresponding information standard should be consulted.

## 2.11 Including "secondary" resources when sending information

---

When the client creates/updates a resource on a server, it often needs to reference and include "secondary" resources which are not the primary focus of the interaction - e.g. an Observation must indicate its subject using a reference to a Patient resource. The situation may arise that the client includes an instance of a resource that already exists on the server, in the transaction (for example, when the client doesn't know about the existing resource). In this case, the server should align these new resources with the ones it already has.

A data conflict may arise if these secondary resources carry details that are not of interest within the particular context of the information standard. For example, the client posts an Observation and sends along a Patient resource containing a different telephone number than the one that is on record at the server side. This change in telephone number is not relevant in this particular context, since the interaction is about creating the Observation.

Unless more specific guidance is given by the information standard, the following guidance applies for dealing with such conflicts:

- Sending secondary resources SHALL only be done when the information standard specifies that a transaction interaction is to be used<sup>[5]</sup>.
- If so, the client MAY send a bare-bones instance as the secondary resource containing just minimal information, but this SHOULD include the information necessary for the server to match its own copy. The information standard MAY specify which information is minimally needed for such a secondary resource and MAY also restrict the information that is sent in the secondary resource.
- The client MUST include a `Bundle.entry.request` for each of the resources in the Bundle, including the secondary resources that are only there for referencing purposes. `.request` SHOULD be a POST to the appropriate endpoint to indicate the create operation. As an alternative, the update operation MAY be used, but this is strongly discouraged; if the client has knowledge on the id of the resource on the server, an absolute URL is strongly preferred.
- If the server can unequivocally deduplicate this secondary resource (for example, because the patient is already known from the security context or the healthcare provider is recognized by the UZI number), it MAY choose to ignore the differences of this resource or it MAY incorporate them (see the section on transactional integrity (<https://hl7.org/fhir/r4/http.html#transactional-integrity>) in the FHIR specification). Please note that absence of information should not result in deletion of information on the server, as the client could send a bare-bones instance. If the server chooses to ignore the differences, it SHOULD do so silently by responding with a 200 OK status code for the corresponding `Bundle.entry.response.status` in the response Bundle.
- The server SHALL rewrite the references in the primary resource to the instance of the secondary resource it already has.

## 2.12 Validation and terminology

---

The outcome of profile based validation can depend on the tooling (validator, terminology server) and its settings used. Implementers should be aware of the following topics regarding terminology when validating instances of nl-core or derived profiles.

### 2.12.1 SNOMED CT - Dutch edition

Nictiz uses codes that are only present in the Dutch edition of SNOMED CT. Validators should take this edition (URI: <http://snomed.info/sct/11000146104>) into account when calling upon a terminology server. Without this edition, you may get a response suitable for the international edition.

### 2.12.2 CodeSystems without concepts defined

The zibs or UCSDs often prescribe the use of 'external' code systems, the contents of which may not be available without proper license. In FHIR, this is manifested by the use of ValueSets that contain `ValueSet.compose.include.system`, without specific `.concepts`, to indicate that all codes from the referenced CodeSystem are allowed. These CodeSystem resources have `CodeSystem.content` set to *not-present*. Some validators give a *warning* when a code from such ValueSet and CodeSystem is used, but some validators may give an *error*.

## 2.13 Missing information

---

The FHIR profile or the the ART-DECOR specification of a use case (or transaction) may require the presence of an element (cardinality 1..1 or 1..\*) while a sending system doesn't have the required data. When this element is marked 'Required' or 'R' ([https://wiki.art-decor.org/index.php?title=DECOR-rules#Mandatory\\_.2F\\_Conformance](https://wiki.art-decor.org/index.php?title=DECOR-rules#Mandatory_.2F_Conformance)) (as opposed to 'Mandatory' or 'M' ([https://wiki.art-decor.org/index.php?title=DECOR-rules#Mandatory\\_.2F\\_Conformance](https://wiki.art-decor.org/index.php?title=DECOR-rules#Mandatory_.2F_Conformance))) in the ART-DECOR specification, the following guidance can be applied. Please note that in the case of elements designated 'Mandatory', the data MUST be present.

1. For *non-coded* data elements, use the Data Absent Reason Extension (<http://hl7.org/fhir/R4/extension-data-absent-reason.html>) with an appropriate code.
2. For *coded* data elements:
  - *Example, preferred, or extensible* binding strengths (CodeableConcept data types):
    - If the source system has text but no coded data, the `.text` element is used. Use the appropriate *other* code from the ValueSet if available.
    - If there is neither text nor coded data:
      - Use the appropriate *unknown* concept code from the ValueSet if available.
      - If the ValueSet does not have the appropriate *unknown* concept code, use the Data Absent Reason Extension (<http://hl7.org/fhir/R4/extension-data-absent-reason.html>) with an appropriate code.
  - *Required* binding strength (CodeableConcept or code data types):
    - If the ValueSet includes the appropriate *other* code and the source system has text but no coded data, use the *other* code in combination with the `.text` element.
    - If there is neither text nor coded data, use the appropriate *unknown* concept code from the ValueSet if available.
    - If the ValueSet does not have an appropriate *other* or *unknown* concept code, you must use a concept from the ValueSet, otherwise the instance will not be conformant.

Profiles do not explicitly define when or how to apply FHIR DataAbsentReason.

## 2.14 Resource .text or "the narrative"

---

The FHIR R4 specification says instances SHOULD contain a human readable summary of the contained data, which can be used as fallback by the receiver. This human readable text is contained in the `.text` element (<https://hl7.org/fhir/R4/domainresource-definitions.html#DomainResource.text>) with data type *Narrative* (<https://hl7.org/fhir/R4/narrative.html>). This human readable text is often referred to as "the narrative".

This implementation guide has the following expectations regarding the use of narratives:

- Senders SHOULD provide a "clinically safe" (see below) narrative of status *extensions* (preferred) or *generated* (see below), unless:
  - The resource does not support narratives. This is the case for resources that do not derive from DomainResource, like Binary.
  - The resource is contained in another resource.
  - It is explicitly documented in the information standard to do otherwise.

- Receivers SHALL support the narrative. Ignoring the narrative is considered passive support. Support for the narrative SHALL be in accordance with the status and explicitly documented use in the relevant information standard.
- Receivers SHALL NOT generically depend on the presence of a clinically safe narrative as their only means to present data to users when an explicitly documented use case for status empty or additional exists. Also, using just the narrative you would not expect to produce graphs, support for medication alerts and other functionality.

### 2.14.1 "Clinically safe" narratives

Regarding the contents of the narrative, the [FHIR R4 spec states \(https://hl7.org/fhir/R4/domainresource-definitions.html#DomainResource.text\)](https://hl7.org/fhir/R4/domainresource-definitions.html#DomainResource.text) that:

The narrative need not encode all the structured data, but is required to contain sufficient detail to make it "clinically safe" for a human to just read the narrative.

And for the Narrative data type:

Structured data SHOULD NOT generally contain information of importance to human readers that is omitted from the narrative. Creators of FHIR resources should not assume that systems will render (or that humans will see) data that is not in the narrative.

The decision to what entails "clinically safe" information, is somewhat arbitrary. Guidelines for "clinically safe" inspired by the [FHIR W5 report \(https://hl7.org/fhir/R4/w5.html\)](https://hl7.org/fhir/R4/w5.html):

- If you only had the narrative: the intention and context should be clear to you.
- Who. Patient/subject, performer, author and/or other primary actors. Is this about family history or the actual patient?
- What. The topic for the resource is clear.
- When. Is this past, present, or future information?
- Why. What led up to the event? E.g. Condition, Referral that caused Encounter or Procedure.
- Where. Location of the event/action in the resource.
- Context. Relevant context as present in the resource. E.g. status, Encounter, EpisodeOfCare, Diagnosis, Condition, BodySite.
- Rendering every identifier and/or code detail may not be necessary. `Resource.text` is for human assessment of the situation so emphasis should be on display text, representing identifier/code/system as relevant in the context.

Informal note: Producing a "clinically safe" narrative can be cumbersome. Various reference frameworks like [HAPI \(https://hapifhir.io/hapi-fhir/docs/model/narrative\\_generation.html\)](https://hapifhir.io/hapi-fhir/docs/model/narrative_generation.html) include a Narrative Generator of some sort.

### 2.14.2 Narrative status

Data type *Narrative* defines different status codes (<https://hl7.org/fhir/R4/valueset-narrative-status.html>) to indicate the origin of the narrative text. This implementation guide expects the most common use case to be the status *extensions*, which means that the narrative is generated entirely from the structured data and the extensions used. The reason for this is that most resources can have extension content defined on them. This status can still be used if a particular instance doesn't contain any extensions.

Alternatively, status *generated* may be used if the sender did not anticipate inclusion of regular extension content in the narrative.

**Note:** for both status *extensions* and *generated*, all modifier extensions SHOULD be included in the narrative as they always affect the semantics.



Senders SHOULD NOT use the other possible statuses unless it is explicitly documented why, e.g. in an information standard. Receivers MAY choose to support narratives with status *additional*, which means that it holds information not contained in the structured data. Receivers - upon rendering narrative with status *additional* - SHOULD make it clear to the user that the information is a fragment, and not the entire thing, in some suitable way.

### 3 Release notes

Release	Version	BITS ticket	Description
1.0 - Maart 2024	1.0.4	<a href="https://bits.nictiz.nl/browse/MM-5114">MM-5114 (https://bits.nictiz.nl/browse/MM-5114)</a>	Spelfouten opgelost en andere kleine aanpassingen doorgevoerd.
1.0 - September 2023	1.0.3	<a href="https://bits.nictiz.nl/browse/MM-4920">MM-4920 (https://bits.nictiz.nl/browse/MM-4920)</a>	In de algemene FHIR IG wordt gesteld dat verplichte elementen de inhoudelijke vulling weg kunnen laten, zolang ze niet mandatory zijn op functioneel niveau. Deze nuance is duidelijker omschreven.
		<a href="https://bits.nictiz.nl/browse/MM-4972">MM-4972 (https://bits.nictiz.nl/browse/MM-4972)</a>	Spelfouten opgelost en andere kleine aanpassingen doorgevoerd.
1.0 - Juni 2022	1.0.2	<a href="https://bits.nictiz.nl/browse/MM-4821">MM-4821 (https://bits.nictiz.nl/browse/MM-4821)</a>	Aan de overkoepelende FHIR-IG is uitleg toegevoegd waaróm een display-waarde bij een reference verplicht/verwacht is. Ook is de opmerking toegevoegd dat de display-waarde niet weggelaten moet worden als deze bv. privacyproblemen zou geven, maar dat hier een expliciete melding verwacht wordt dat deze gemaskeerd is.
1.0 - December 2022	1.0.1	<a href="https://bits.nictiz.nl/browse/MM-3839">MM-3839 (https://bits.nictiz.nl/browse/MM-3839)</a>	De sectie over het meesturen van "secundaire" resources in de implementatiegidsen voor FHIR STU3 en R4 beargumenteert dat het updaten van bestaande resources ontmoedigd wordt, maar dat wordt verwoord als "PUT", wat breder inzetbaar is dan een update. Dit is aangepast naar "de update operation".
1	1.0.0	<a href="https://bits.nictiz.nl/browse/MM-2856">MM-2856 (https://bits.nictiz.nl/browse/MM-2856)</a>	First release of the Nictiz FHIR IG R4.

1. Although the canonical URI is not formally defined in the ValueSet, in practical terms it can be found here.
2. For an overview of URLs defined, see the FHIR core specification (<https://www.hl7.org/fhir/R4/terminologies-systems.html>) or HL7 NL for the Dutch realm ([http://hl7.nl/wiki/index.php?title=OIDs\\_en\\_FHIR\\_System\\_URIs](http://hl7.nl/wiki/index.php?title=OIDs_en_FHIR_System_URIs)).
3. Some implementations may choose to populate the `.id` element with the unprefixd UUID or OID in cases when the `.id` field is expected, for example in searchset Bundles. The FHIR specification is ambiguous on the requirements to do so. For a long discussion on the topic, see chat.fhir.org (<https://chat.fhir.org/#narrow/stream/179166-implementers/topic/Confused.20about.20URN.20fullUrl's.20and.20resource.20Eid's>).
4. The use of OIDs as id has some limitations:
  - The `.id` has a maximum length of 64 characters, so only OIDs that contain less than 64 characters can be

used.

- OID-based ids are expected to be stable, just like RESTful ids. Different Bundle instances SHALL always use the same OID-based ids.
- Some *objects* may have multiple OID-based identifiers, like AGB and UZI for practitioners. Which one do you choose? And what happens when the UZI number becomes available after you've already used the AGB as `.id`?
- Specifically for HL7 V3 based backends: an HL7 V3 II data type normally consists of a `@root` (OID) and an `@extension` (string). Concatenation of `@root` and `@extension` only yields a new OID if `@extension` is numeric. Other combinations of `@root.@extension`, may yield a valid `.id` (pattern `[A-Za-z0-9\-\.\ ]{1,64}`), but may not be a valid OID. In this case it is no longer possible to create a `.fullUrl` consistent with the `.id` based on the `urn:oid: scheme`.

5. Note that a batch interaction SHALL not have interdependencies between resources (<https://hl7.org/fhir/r4/http.html#transaction>) and therefore by definition is not suitable for use cases where secondary resources can be expected.

---

Overgenomen van "[https://informatiestandaarden.nictiz.nl/index.php?title=FHIR:V1.0\\_FHIR\\_IG\\_R4&oldid=212047](https://informatiestandaarden.nictiz.nl/index.php?title=FHIR:V1.0_FHIR_IG_R4&oldid=212047)"

---

**Deze pagina is voor het laatst bewerkt op 25 mrt 2024 om 21:31.**

De inhoud is beschikbaar onder de [Creative Commons Naamsvermelding-Gelijk delen](#) tenzij anders aangegeven.