2.1.6.4 JSON Representation of Resources

|  |  |  |
| --- | --- | --- |
| [Implementable Technology Specifications [icon](http://www.hl7.org/special/committees/xml/index.cfm)](http://www.hl7.org/special/committees/xml/index.cfm) Work Group | [Maturity Level](https://hl7.org/fhir/versions.html#maturity): Normative | [Standards Status](https://hl7.org/fhir/versions.html#std-process): [Normative](https://hl7.org/fhir/versions.html#std-process) |

The JSON representation for a resource is based on the [JSON format described in STD 90 (RFC 8259) [icon](https://www.rfc-editor.org/info/std90)](https://www.rfc-editor.org/info/std90), and is described using this format:

{

"resourceType" : "[**[Resource Type]**](https://hl7.org/fhir/resourcelist.html)",

// from [Source](https://hl7.org/fhir/json.html): [property0](https://hl7.org/fhir/json.html)

"[property1](https://hl7.org/fhir/json.html)" : "<[[primitive]](https://hl7.org/fhir/datatypes.html)>", // short description

"[property2](https://hl7.org/fhir/json.html)" : { [[Datatype]](https://hl7.org/fhir/datatypes.html) }, // short description

"[property3](https://hl7.org/fhir/json.html)" : { // Short Description

"[propertyA](https://hl7.org/fhir/json.html)" : { [CodeableConcept](https://hl7.org/fhir/datatypes.html#CodeableConcept) }, // [Short Description](https://hl7.org/fhir/json.html) ([Example](https://hl7.org/fhir/terminologies.html#example))

},

"[property4](https://hl7.org/fhir/json.html)" : [{ // Short Description

"[propertyB](https://hl7.org/fhir/json.html)" : { [Reference](https://hl7.org/fhir/references.html#Reference)([ResourceType](https://hl7.org/fhir/resourcelist.html)) } // **R!** Short Description

}]

}

Using this format:

* To build a valid JSON instance of a resource, replace the contents of the property values with valid content as described by the type rules and content description found in the property value for each element
* In this example:
  1. property1 has a primitive datatype; the value of the property will be as described for the stated type
  2. property2 has a complex datatype; the value of the property is an object that has the content as described for the stated type
  3. property3 is an object property that contains additional properties (e.g. propertyA; the allowable properties are listed, and also include extensions as appropriate)
  4. property4 is an array property that contains items which are objects themselves. The items may have any of the types already encountered in points 1-3
  5. propertyA is an example of an object property that has a binding to a value set - the Short description is a link to the value set. In addition, the binding strength is shown
  6. propertyB is an example of an object property that has a reference to a particular kind of resource
* Property names are case-sensitive (though duplicates that differ only in case are never defined)
* Property names SHALL be unique. Note: this is not explicitly stated in the original JSON specification,so stated for clarity here
* Properties can appear in any order
* XHTML is represented as an escaped string
* Objects are never empty. If an element is present in the resource, it SHALL have properties as defined for its type, or 1 or more [extensions](https://hl7.org/fhir/extensibility.html)
* String property values can never be empty. Either the property is absent, or it is present with at least one character of content
* The **R!** denotes that an element is mandatory - it must be present (or in an array, at least one item must be present)
* In this format, // is used for comments. While // is legal in Javascript, it is not legal in JSON, and comments SHALL not be in JSON instances irrespective of whether particular applications ignore them
* The character encoding is always UTF-8
* The MIME-type for this format is application/fhir+json.

Given the way [extensions](https://hl7.org/fhir/extensibility.html) work, applications reading JSON resources will never encounter unknown properties. However, once an application starts trading with other applications that conform to later versions of this specification, unknown properties may be encountered. Applications MAY choose to ignore unknown properties in order to foster forwards compatibility in this regard, but may also choose not to.

This page contains some XML examples for the purposes of comparison between the two formats. The formats page has a [comparison between the JSON and XML formats](https://hl7.org/fhir/resource-formats.html#comparison).

2.1.6.4.1 JSON Representation for repeating elements

An element that has a maximum cardinality of >1 (e.g. x..\* in the definitions) may occur more than once in the instance. In JSON, this is done by using an array type. Note that:

* The name of the array is singular
* An item that may repeat is represented as an array even in the case that it doesn't repeat so that the process of parsing the resource is the same either way

So a [CodeableConcept](https://hl7.org/fhir/datatypes.html#CodeableConcept) is represented in JSON like this:

{

"coding": [

{

"system" : "http://snomed.info/sct",

"code" : "104934005"

},

{

"system" : "http://loinc.org",

"code" : "2947-0"

}

]

}

XML for comparison:

<code>

<coding>

<system value="http://snomed.info/sct"/>

<code value="104934005"/>

</coding>

<coding>

<system value="http://loinc.org"/>

<code value="2947-0"/>

</coding>

</code>

2.1.6.4.2 JSON representation of primitive elements

FHIR elements with primitive datatypes are represented in two parts:

* A JSON property with the name of the element, which has a JSON type of number, boolean, or string
* a JSON property with \_ prepended to the name of the element, which, if present, contains the value's id and/or extensions

The FHIR types [integer](https://hl7.org/fhir/datatypes.html#integer), [unsignedInt](https://hl7.org/fhir/datatypes.html#unsignedInt), [positiveInt](https://hl7.org/fhir/datatypes.html#positiveInt) and [decimal](https://hl7.org/fhir/datatypes.html#decimal) are represented as a JSON number, the FHIR type [boolean](https://hl7.org/fhir/datatypes.html#boolean) as a JSON boolean, and all other types (including [integer64](https://hl7.org/fhir/datatypes.html#integer64)) are represented as a JSON string which has the same content as that specified for the relevant datatype. Whitespace is always significant (i.e. no leading and trailing spaces for non-strings).

"code" : "abc",

"birthDate" : "1972-11-30",

"deceased" : false,

"count" : 23

}

For comparison, this is represented in XML as

<**code** value="abc"/> <!-- code -->

<**birthDate** value="1972-11-30"/> <!-- dateTime -->

<**deceased** value="false" /> <!-- boolean -->

<**count** value="23" /> <!-- integer -->



When using a JavaScript JSON.parse() implementation, note that JavaScript natively supports only one numeric datatype, which is a floating point number. This can cause loss of precision for FHIR numbers. In particular, trailing 0s after a decimal point will be lost e.g. 2.00 will be converted to 2. The FHIR decimal datatype is defined such that precision, including trailing zeros, is preserved for presentation purposes, and this is widely regarded as critical for correct presentation of clinical measurements. Implementations should consider using a custom parser and big number library (e.g. [https://github.com/jtobey/javascript-bignum [icon](https://github.com/jtobey/javascript-bignum)](https://github.com/jtobey/javascript-bignum)) to meet these requirements. See also [the precision extension](http://hl7.org/fhir/extensions/StructureDefinition-quantity-precision.html).

If the value has an id attribute, or extensions, then this is represented as follows:

"birthDate": "1970-03-30",

"\_birthDate": {

"id": "314159",

"extension" : [ {

"url" : "http://example.org/fhir/StructureDefinition/text",

"valueString" : "Easter 1970"

}]

}

Note: If the primitive has an id attribute or extension, but no value, only the property with the \_ is rendered.

This is represented in XML as:

<**birthDate** id="314159" value="1970-03-30" >

<extension url="http://example.org/fhir/StructureDefinition/text">

<valueString value="Easter 1970"/>

</extension>

</**birthDate**>

In the case where the primitive element may repeat, it is represented in two arrays. JSON null values are used to fill out both arrays so that the id and/or extension are aligned with the matching value in the first array, as demonstrated in this example:

<**code** value="au"/>

<**code** value="nz">

<extension url="http://hl7.org/fhir/StructureDefinition/display">

<valueString value="New Zealand a.k.a Kiwiland"/>

</extension>

</**code**>

is represented in JSON as:

"code": [ "au", "nz" ],

"\_code": [

null,

{

"extension" : [ {

"url" : "http://hl7.org/fhir/StructureDefinition/display",

"valueString" : "New Zealand a.k.a Kiwiland"

}]

}

]

Note: when one of the repeating elements has no value, it is represented in the first array using a null. When an element has a value but no extension/id, the second array will have a null at the position of that element. If the length of a JSON "element" array is different from the length of its JSON "\_element" array, implementations SHALL infer null values in the suffix of the shorter array. If an array has only null elements, implementations SHOULD omit it entirely.

**Implementation Note:** The representation of primitive datatypes has been split into two parts like this in order to simplify the representation of simple primitive values without id or extensions. This does have the cost of making the representation of the id attribute and extensions more ungainly, but these are both rarely used with primitive datatypes.

2.1.6.4.3 JSON representation of Complex Datatypes and XHTML

Complex [datatypes](https://hl7.org/fhir/datatypes.html) (types that contain named elements of other types) are represented using a JSON object that contains a property for each element in the datatype. Complex datatypes can have ids, which are converted to JSON properties, in the same manner as described for primitives. For example:

<Patient>

<text>

<status value="generated" />

<div xmlns="http://www.w3.org/1999/xhtml"><p>...</p></div>

</text>

<name id="f2">

<use value="official" />

<given value="Karen" />

<family id="a2" value="Van" />

</name>

</Patient>

is represented in JSON as:

{

"resourceType" : "Patient",

"name" : [{

"id" : "f2",

"use" : "official" ,

"given" : [ "Karen" ],

"family" : "Van",

"\_family" : {"id" : "a2"}

}],

"text" : {

"status" : "generated" ,

"div" : "<div xmlns=\"http://www.w3.org/1999/xhtml\"><p>...</p></div>"

}

}

Things to note here are:

* given is a repeating element, so it is serialized as an Array, whether or not it repeats in this instance
* In the family part of name, the id is added represented in \_family as described above, while the id on the name itself is represented as just another property
* The order of JSON properties may differ from the order in the specification
* The XHTML content in the div element which is in the Narrative element text is represented as an escaped string in the value property in JSON. The xhtml root element SHALL be a <div> in the xhtml namespace

2.1.6.4.4 JSON representation of Resources

A resource is a JSON object with a property resourceType which informs the parser which resource type this is:

{

"resourceType" : "Patient",

"text" : {

"status" : "generated" ,

"div" : "<div xmlns=\"http://www.w3.org/1999/xhtml\"><p>...</p></div>"

}

// etc...

}

Note that parsers cannot assume that the resourceType property will come first.

**Implementation Note:** This is a problem for several JSON -> Object serializers that assume that the resourceType property does come first, including [Json.NET [icon](http://james.newtonking.com/json)](http://james.newtonking.com/json). However, some JSON generators do not give the authoring application control of the order of the property values, and so these implementations cannot interoperate with implementations that make assumptions about order. Given that JSON says that the property values are an unordered map of name/value pairs, this specification cannot require that properties come in any particular order, though implementers may choose to fix the property order if they are able (and the reference platforms provided with this specification do so).

There is [a sample file](https://hl7.org/fhir/json-edge-cases.json) with many edge cases to help test JSON parsers.

2.1.6.4.5 Canonical JSON

Resources and/or Bundles may be digitally signed (see [Bundle](https://hl7.org/fhir/bundle.html) and [Provenance](https://hl7.org/fhir/provenance.html)).

This specification defines the following method for canonicalizing FHIR resources, when represented as JSON. The signed set of fragments SHALL be transformed such that:

* No whitespace other than single spaces in property values and in the xhtml in the [Narrative](https://hl7.org/fhir/narrative.html)
* Properties are ordered alphabetically within each object
* Multiple fragments are concatenated with no intervening white-space in the order defined by the element with the [Signature](https://hl7.org/fhir/datatypes.html#Signature) datatype

This canonicalization method is identified by the URI http://hl7.org/fhir/canonicalization/json. The following additional canonicalization URIs are also defined:

|  |  |
| --- | --- |
| http://hl7.org/fhir/canonicalization/json#data | The narrative (Resource.text) is omitted prior to signing (note the deletion is at Resource.text, not Resource.text.div) |
| http://hl7.org/fhir/canonicalization/json#static | In addition to narrative (Resource.text), the Resource.meta element is removed. This makes the signature robust as the content is moved from server to server, or workflow and access tags are added or removed. Note that workflow and security tags may contain information important to the handling of the resource, so meta elements should be protected from tampering by other means if unsigned. |
| http://hl7.org/fhir/canonicalization/json#narrative | This method only retains the Resource.id and Narrative elements |
| http://hl7.org/fhir/canonicalization/json#document | The signs everything in a Bundle, except for the Bundle.id and Bundle.metadata on the root Bundle (allows for a document to be copied from server to server) |

These canonicalization methods allow system the flexibility to sign the various portions of the resource that matter for the workflow the signature serves. These canonicalization algorithms do not work for enveloped signatures. This will be researched and addressed in a future release. This specification may define additional canonicalizations in the future, and other specifications might also define additional canonicalization methods.

**Implementation Note:** One consequence of signing the document is that URLs, identifiers and internal references are frozen and cannot be changed. This might be a desired feature, but it may also cripple interoperability between closed ecosystems where [re-identification](https://hl7.org/fhir/managing.html) frequently occurs. For this reason, it is recommended that systems consider carefully the impact of any signature processes. The impact of signatures on [Document bundles](https://hl7.org/fhir/documents.html) and their related processes is the most well understood use of digital signatures.

2.1.6.4 JSON Representation of Resources

|  |  |  |
| --- | --- | --- |
| [Implementable Technology Specifications [icon](http://www.hl7.org/special/committees/xml/index.cfm)](http://www.hl7.org/special/committees/xml/index.cfm) Work Group | [Maturity Level](https://hl7.org/fhir/versions.html#maturity): Normative | [Standards Status](https://hl7.org/fhir/versions.html#std-process): [Normative](https://hl7.org/fhir/versions.html#std-process) |

The JSON representation for a resource is based on the [JSON format described in STD 90 (RFC 8259) [icon](https://www.rfc-editor.org/info/std90)](https://www.rfc-editor.org/info/std90), and is described using this format:

{

"resourceType" : "[**[Resource Type]**](https://hl7.org/fhir/resourcelist.html)",

// from [Source](https://hl7.org/fhir/json.html): [property0](https://hl7.org/fhir/json.html)

"[property1](https://hl7.org/fhir/json.html)" : "<[[primitive]](https://hl7.org/fhir/datatypes.html)>", // short description

"[property2](https://hl7.org/fhir/json.html)" : { [[Datatype]](https://hl7.org/fhir/datatypes.html) }, // short description

"[property3](https://hl7.org/fhir/json.html)" : { // Short Description

"[propertyA](https://hl7.org/fhir/json.html)" : { [CodeableConcept](https://hl7.org/fhir/datatypes.html#CodeableConcept) }, // [Short Description](https://hl7.org/fhir/json.html) ([Example](https://hl7.org/fhir/terminologies.html#example))

},

"[property4](https://hl7.org/fhir/json.html)" : [{ // Short Description

"[propertyB](https://hl7.org/fhir/json.html)" : { [Reference](https://hl7.org/fhir/references.html#Reference)([ResourceType](https://hl7.org/fhir/resourcelist.html)) } // **R!** Short Description

}]

}

Using this format:

* To build a valid JSON instance of a resource, replace the contents of the property values with valid content as described by the type rules and content description found in the property value for each element
* In this example:
  1. property1 has a primitive datatype; the value of the property will be as described for the stated type
  2. property2 has a complex datatype; the value of the property is an object that has the content as described for the stated type
  3. property3 is an object property that contains additional properties (e.g. propertyA; the allowable properties are listed, and also include extensions as appropriate)
  4. property4 is an array property that contains items which are objects themselves. The items may have any of the types already encountered in points 1-3
  5. propertyA is an example of an object property that has a binding to a value set - the Short description is a link to the value set. In addition, the binding strength is shown
  6. propertyB is an example of an object property that has a reference to a particular kind of resource
* Property names are case-sensitive (though duplicates that differ only in case are never defined)
* Property names SHALL be unique. Note: this is not explicitly stated in the original JSON specification,so stated for clarity here
* Properties can appear in any order
* XHTML is represented as an escaped string
* Objects are never empty. If an element is present in the resource, it SHALL have properties as defined for its type, or 1 or more [extensions](https://hl7.org/fhir/extensibility.html)
* String property values can never be empty. Either the property is absent, or it is present with at least one character of content
* The **R!** denotes that an element is mandatory - it must be present (or in an array, at least one item must be present)
* In this format, // is used for comments. While // is legal in Javascript, it is not legal in JSON, and comments SHALL not be in JSON instances irrespective of whether particular applications ignore them
* The character encoding is always UTF-8
* The MIME-type for this format is application/fhir+json.

Given the way [extensions](https://hl7.org/fhir/extensibility.html) work, applications reading JSON resources will never encounter unknown properties. However, once an application starts trading with other applications that conform to later versions of this specification, unknown properties may be encountered. Applications MAY choose to ignore unknown properties in order to foster forwards compatibility in this regard, but may also choose not to.

This page contains some XML examples for the purposes of comparison between the two formats. The formats page has a [comparison between the JSON and XML formats](https://hl7.org/fhir/resource-formats.html#comparison).

2.1.6.4.1 JSON Representation for repeating elements

An element that has a maximum cardinality of >1 (e.g. x..\* in the definitions) may occur more than once in the instance. In JSON, this is done by using an array type. Note that:

* The name of the array is singular
* An item that may repeat is represented as an array even in the case that it doesn't repeat so that the process of parsing the resource is the same either way

So a [CodeableConcept](https://hl7.org/fhir/datatypes.html#CodeableConcept) is represented in JSON like this:

{

"coding": [

{

"system" : "http://snomed.info/sct",

"code" : "104934005"

},

{

"system" : "http://loinc.org",

"code" : "2947-0"

}

]

}

XML for comparison:

<code>

<coding>

<system value="http://snomed.info/sct"/>

<code value="104934005"/>

</coding>

<coding>

<system value="http://loinc.org"/>

<code value="2947-0"/>

</coding>

</code>

2.1.6.4.2 JSON representation of primitive elements

FHIR elements with primitive datatypes are represented in two parts:

* A JSON property with the name of the element, which has a JSON type of number, boolean, or string
* a JSON property with \_ prepended to the name of the element, which, if present, contains the value's id and/or extensions

The FHIR types [integer](https://hl7.org/fhir/datatypes.html#integer), [unsignedInt](https://hl7.org/fhir/datatypes.html#unsignedInt), [positiveInt](https://hl7.org/fhir/datatypes.html#positiveInt) and [decimal](https://hl7.org/fhir/datatypes.html#decimal) are represented as a JSON number, the FHIR type [boolean](https://hl7.org/fhir/datatypes.html#boolean) as a JSON boolean, and all other types (including [integer64](https://hl7.org/fhir/datatypes.html#integer64)) are represented as a JSON string which has the same content as that specified for the relevant datatype. Whitespace is always significant (i.e. no leading and trailing spaces for non-strings).

"code" : "abc",

"birthDate" : "1972-11-30",

"deceased" : false,

"count" : 23

}

For comparison, this is represented in XML as

<**code** value="abc"/> <!-- code -->

<**birthDate** value="1972-11-30"/> <!-- dateTime -->

<**deceased** value="false" /> <!-- boolean -->

<**count** value="23" /> <!-- integer -->



When using a JavaScript JSON.parse() implementation, note that JavaScript natively supports only one numeric datatype, which is a floating point number. This can cause loss of precision for FHIR numbers. In particular, trailing 0s after a decimal point will be lost e.g. 2.00 will be converted to 2. The FHIR decimal datatype is defined such that precision, including trailing zeros, is preserved for presentation purposes, and this is widely regarded as critical for correct presentation of clinical measurements. Implementations should consider using a custom parser and big number library (e.g. [https://github.com/jtobey/javascript-bignum [icon](https://github.com/jtobey/javascript-bignum)](https://github.com/jtobey/javascript-bignum)) to meet these requirements. See also [the precision extension](http://hl7.org/fhir/extensions/StructureDefinition-quantity-precision.html).

If the value has an id attribute, or extensions, then this is represented as follows:

"birthDate": "1970-03-30",

"\_birthDate": {

"id": "314159",

"extension" : [ {

"url" : "http://example.org/fhir/StructureDefinition/text",

"valueString" : "Easter 1970"

}]

}

Note: If the primitive has an id attribute or extension, but no value, only the property with the \_ is rendered.

This is represented in XML as:

<**birthDate** id="314159" value="1970-03-30" >

<extension url="http://example.org/fhir/StructureDefinition/text">

<valueString value="Easter 1970"/>

</extension>

</**birthDate**>

In the case where the primitive element may repeat, it is represented in two arrays. JSON null values are used to fill out both arrays so that the id and/or extension are aligned with the matching value in the first array, as demonstrated in this example:

<**code** value="au"/>

<**code** value="nz">

<extension url="http://hl7.org/fhir/StructureDefinition/display">

<valueString value="New Zealand a.k.a Kiwiland"/>

</extension>

</**code**>

is represented in JSON as:

"code": [ "au", "nz" ],

"\_code": [

null,

{

"extension" : [ {

"url" : "http://hl7.org/fhir/StructureDefinition/display",

"valueString" : "New Zealand a.k.a Kiwiland"

}]

}

]

Note: when one of the repeating elements has no value, it is represented in the first array using a null. When an element has a value but no extension/id, the second array will have a null at the position of that element. If the length of a JSON "element" array is different from the length of its JSON "\_element" array, implementations SHALL infer null values in the suffix of the shorter array. If an array has only null elements, implementations SHOULD omit it entirely.

**Implementation Note:** The representation of primitive datatypes has been split into two parts like this in order to simplify the representation of simple primitive values without id or extensions. This does have the cost of making the representation of the id attribute and extensions more ungainly, but these are both rarely used with primitive datatypes.

2.1.6.4.3 JSON representation of Complex Datatypes and XHTML

Complex [datatypes](https://hl7.org/fhir/datatypes.html) (types that contain named elements of other types) are represented using a JSON object that contains a property for each element in the datatype. Complex datatypes can have ids, which are converted to JSON properties, in the same manner as described for primitives. For example:

<Patient>

<text>

<status value="generated" />

<div xmlns="http://www.w3.org/1999/xhtml"><p>...</p></div>

</text>

<name id="f2">

<use value="official" />

<given value="Karen" />

<family id="a2" value="Van" />

</name>

</Patient>

is represented in JSON as:

{

"resourceType" : "Patient",

"name" : [{

"id" : "f2",

"use" : "official" ,

"given" : [ "Karen" ],

"family" : "Van",

"\_family" : {"id" : "a2"}

}],

"text" : {

"status" : "generated" ,

"div" : "<div xmlns=\"http://www.w3.org/1999/xhtml\"><p>...</p></div>"

}

}

Things to note here are:

* given is a repeating element, so it is serialized as an Array, whether or not it repeats in this instance
* In the family part of name, the id is added represented in \_family as described above, while the id on the name itself is represented as just another property
* The order of JSON properties may differ from the order in the specification
* The XHTML content in the div element which is in the Narrative element text is represented as an escaped string in the value property in JSON. The xhtml root element SHALL be a <div> in the xhtml namespace

2.1.6.4.4 JSON representation of Resources

A resource is a JSON object with a property resourceType which informs the parser which resource type this is:

{

"resourceType" : "Patient",

"text" : {

"status" : "generated" ,

"div" : "<div xmlns=\"http://www.w3.org/1999/xhtml\"><p>...</p></div>"

}

// etc...

}

Note that parsers cannot assume that the resourceType property will come first.

**Implementation Note:** This is a problem for several JSON -> Object serializers that assume that the resourceType property does come first, including [Json.NET [icon](http://james.newtonking.com/json)](http://james.newtonking.com/json). However, some JSON generators do not give the authoring application control of the order of the property values, and so these implementations cannot interoperate with implementations that make assumptions about order. Given that JSON says that the property values are an unordered map of name/value pairs, this specification cannot require that properties come in any particular order, though implementers may choose to fix the property order if they are able (and the reference platforms provided with this specification do so).

There is [a sample file](https://hl7.org/fhir/json-edge-cases.json) with many edge cases to help test JSON parsers.

2.1.6.4.5 Canonical JSON

Resources and/or Bundles may be digitally signed (see [Bundle](https://hl7.org/fhir/bundle.html) and [Provenance](https://hl7.org/fhir/provenance.html)).

This specification defines the following method for canonicalizing FHIR resources, when represented as JSON. The signed set of fragments SHALL be transformed such that:

* No whitespace other than single spaces in property values and in the xhtml in the [Narrative](https://hl7.org/fhir/narrative.html)
* Properties are ordered alphabetically within each object
* Multiple fragments are concatenated with no intervening white-space in the order defined by the element with the [Signature](https://hl7.org/fhir/datatypes.html#Signature) datatype

This canonicalization method is identified by the URI http://hl7.org/fhir/canonicalization/json. The following additional canonicalization URIs are also defined:

|  |  |
| --- | --- |
| http://hl7.org/fhir/canonicalization/json#data | The narrative (Resource.text) is omitted prior to signing (note the deletion is at Resource.text, not Resource.text.div) |
| http://hl7.org/fhir/canonicalization/json#static | In addition to narrative (Resource.text), the Resource.meta element is removed. This makes the signature robust as the content is moved from server to server, or workflow and access tags are added or removed. Note that workflow and security tags may contain information important to the handling of the resource, so meta elements should be protected from tampering by other means if unsigned. |
| http://hl7.org/fhir/canonicalization/json#narrative | This method only retains the Resource.id and Narrative elements |
| http://hl7.org/fhir/canonicalization/json#document | The signs everything in a Bundle, except for the Bundle.id and Bundle.metadata on the root Bundle (allows for a document to be copied from server to server) |

These canonicalization methods allow system the flexibility to sign the various portions of the resource that matter for the workflow the signature serves. These canonicalization algorithms do not work for enveloped signatures. This will be researched and addressed in a future release. This specification may define additional canonicalizations in the future, and other specifications might also define additional canonicalization methods.

**Implementation Note:** One consequence of signing the document is that URLs, identifiers and internal references are frozen and cannot be changed. This might be a desired feature, but it may also cripple interoperability between closed ecosystems where [re-identification](https://hl7.org/fhir/managing.html) frequently occurs. For this reason, it is recommended that systems consider carefully the impact of any signature processes. The impact of signatures on [Document bundles](https://hl7.org/fhir/documents.html) and their related processes is the most well understood use of digital signatures.