This is a REVISED proposal being considered for the W3C DID Core Specification (or an associated Note). The original proposal (now deprecated) is here. Please direct all comments to the DID Core issue thread.

Appendix A: What Does a DID Identify?

Although the DID specification clearly states that *a DID identifies the DID subject*, and that a DID subject can be anything that can be identified with a URI, some finer details of DID identification architecture can be important to certain applications, particularly those for the Semantic Web. This appendix explains these finer details.

Information resources and non-information resources

In W3C Semantic Web architecture, all resources have URIs, but there is a distinction between **information resources** and **non-information resources**. An information resource is any type of digital object. In most (but not all) cases, such an object has one or more **representations** that can be retrieved over a digital network. Examples include web pages, files, images, videos, audio clips, and any other data structure that can be "on" the World Wide Web.

A non-information resource is anything that cannot directly be "on" the World Wide Web because it does not exist as a digital object on a network. Examples include people, organizations, physical objects (the Eiffel Tower, the moon), places ("Bolivia", "Gibraltar"), or logical concepts ("calculus", "peace", "unicorn").

Such non-information resources **cannot have representations** because by definition they are not "made" of information. However, they still need to be referred to by people (and by other information resources). For this reason, the W3C recommends

1. **A non-information resource should still be identified with a URI** that uniquely identifies that resource (and nothing else).

2. **That URI should enable discovery of other URIs** that identify other information resources associated with the non-information resource.
3. **Those other information resources can return *descriptions*** of the non-information resource. A description does not *represent* a resource, it only *describes* it.

These architectural requirements posed a dilemma for the Semantic Web community that can be summarized by the following recommendation from the W3C:[1]

> *There should be no confusion between identifiers for Web documents and identifiers for other resources. URIs are meant to identify only one of them, so one URI can't stand for both a Web document and a real-world object.*

In short, the dilemma is: how do you distinguish between URIs for information resources that *should* return one or more representations and URIs for non-information resources that *must not* return any representations—yet *should* still be able to be dereferenced to discover other URIs for associated information resources.[2]

[1] "Cool URIs for the Semantic Web": https://www.w3.org/TR/cooluris/
[2] In Semantic Web architecture this is known as the HTTP Range 14 problem.

The answer the W3C Technical Architecture Board (TAB) arrived at for this dilemma can be summarized as follows:[3]

1. A URI always identifies one resource and only one resource—no matter whether it is an information resource or a non-information resource.
2. If the resource is an information resource, the URI can be used to directly retrieve representations of the resource.
3. If the resource is a non-information resource, the URI identifying it must not return any representations of the resource. However it should be possible to use that URI to discover other URIs that identify *associated* information resources. Those information resources can return *descriptions* of the non-information resource.
4. This mapping between the URI identifying the non-information resource and other URIs identifying associated information resources can be accomplished in one of two ways:
    a. **The URI for the non-information resource can be constructed by adding a URI fragment to an information resource URI.** In this case, only the information resource URI can be used to retrieve a description of the non-information resource. The W3C TAB calls this option “hash URIs”.
    b. **An HTTP server can be programmed to provide a HTTP 303 `seeOther` response code in response to a request for the non-information resource URI**. This 303 response code can provide a reference to an associated information resource URI. The W3C TAB calls this option “303 URIs”.

While both of these solutions work, the Semantic Web community has never been entirely happy with either. In the first case, some information resources use URI fragments to identify other information resources, so you cannot assume that all URIs ending in fragments identify non-information resources. In the second case, requiring an HTTP request to determine the semantic status of a URI generates extra web traffic, and in some cases URIs identifying non-information resources simply do not have an associated web server.

[3] “Cool URIs for the Semantic Web”: https://www.w3.org/TR/cooluris/

How DID architecture addresses this challenge

DIDs offer a different solution to this dilemma. Here's how it works.

First, by definition, **the DID subject identified by a DID is always a non-information resource**. Even if the DID refers to a web page, for example, the DID itself does not refer to the URL of the web page that can be dereferenced at a particular point in time ("the weather in Seattle today"), but rather to the abstract concept of what that web page represents, (e.g., "the Seattle weather page"). Thus DIDs are by definition *abstract identifiers* and DID subjects enable *abstract identification* of any kind of resource—information or non-information.

Secondly, because the DID subject is by definition a non-information resource, the DID resolution function returns **a description of the DID subject in the form of a DID document**. As described in the W3C's Cool URIs for the Semantic Web, this DID document is never considered a *representation* of the DID subject because a non-information resource does not have any direct representation, only descriptions. This is illustrated in Figure 1 below.

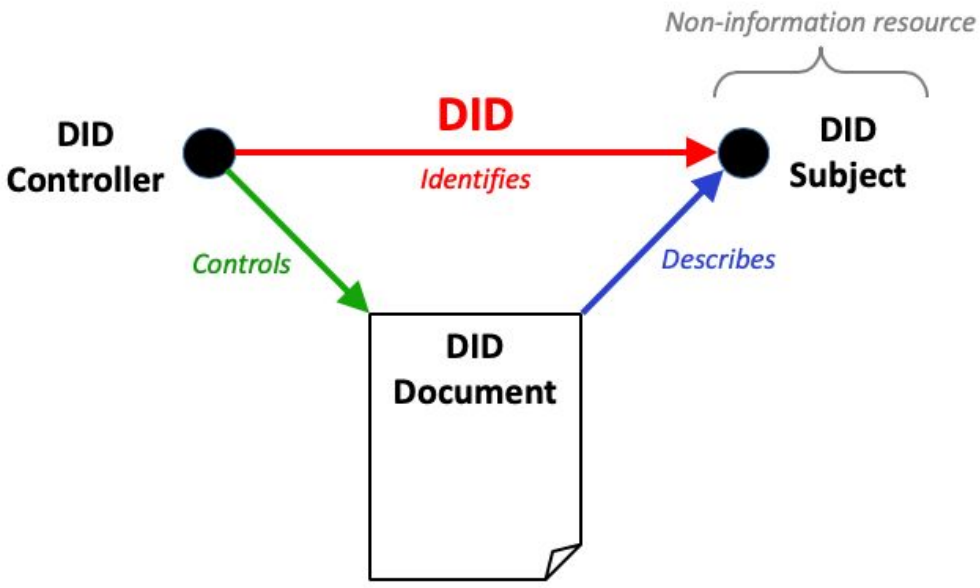

Figure 1: A DID identifies a DID subject as a non-information resource; it resolves to a DID document that describes this non-information resource

So now the question becomes: what is the nature of the resource for which the DID subject is an abstraction? There are two cases:

1. **CASE 1: The DID subject is an abstraction of a non-information resource.** This means the DID is an abstract identifier for a person, organization, or anything else that is NOT inherently a digital object and does NOT live on a digital network.
2. **CASE 2: The DID subject is an abstraction of an information resource.** This means

the DID is an abstract identifier for a web page, file, image, video, audio clip, or any other data structure that can be “on” the World Wide Web.

In both these cases, the DID document can **point to other information resources that further describe the DID subject.** But how this is accomplished depends on whether it is Case 1 or Case 2.

CASE 1: Describing non-information resources with the `service` or `seeOther` properties

As explained above, the W3C recommends that the URI for a non-information resource should not return a representation because that would confuse it with an information resource. A non-information resource can only be *described* by an information resource. DID abstract identification architecture **standardizes this description in the form of a DID document**.

In this way, DID architecture already fulfills this recommendation from the W3C:[4]

> *Given only a URI, machines and people should be able to retrieve a description about the resource identified by the URI from the Web. Such a look-up mechanism is important to establish shared understanding of what a URI identifies. Machines should get RDF data and humans should get a readable representation, such as HTML. The standard Web transfer protocol, HTTP, should be used.*

In a DID document, a DID controller has two options for how to describe an information resource associated with DID subject that is an abstraction of a non-information resource:

1. Use the `seeOther` property to describe how to access other information about the DID subject. See the example below.
2. Use the `service` property to describe how to interact with the DID subject. See section ___ of DID Core.

An example of the `seeOther` property is shown in this JSON-LD representation of a DID document:

```
{
  "@context": "https://www.w3.org/ns/did/v1",
  "id": "did:example:123456789abcdefghi",
  "seeOther": [
    "https://example.com/some/other/description.html",
    "https://example.com/some/other/description.rdf"
   ]
}
```

[4] “Cool URIs for the Semantic Web”: https://www.w3.org/TR/cooluris/. Note that, although it is not strictly required that a DID document use an RDF-based representation such as JSON-LD, it certainly can use that format to meet the letter of this W3C recommendation.

The use of the `seeOther` property is visually illustrated in Figure 2:

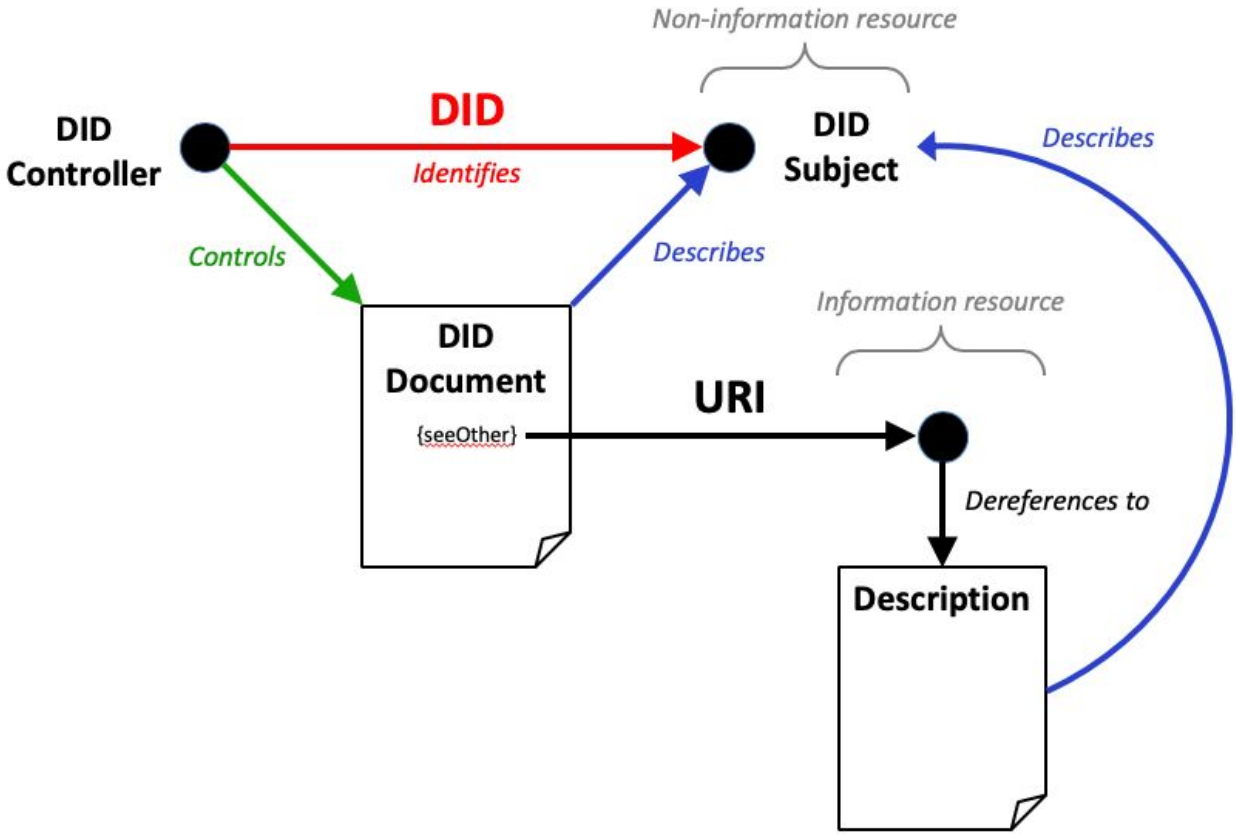

Figure 2: A DID document using a seeOther property to reference an information resource further describing a DID subject as a non-information resource

CASE 2: Describing information resources with the `representation` property

The key difference between non-information resources and information resources is that **only the latter can directly return representations**. So if a DID subject is an abstraction of an information resource, the DID controller can use the `representation` property of a DID document to precisely map the DID to URIs for representations of the information resource.

There are two options for doing this—one by value and one by reference. This JSON-LD provides an example of both:

```
{
  "@context": "https://www.w3.org/ns/did/v1",
  "id": "did:example:123456789abcdefghi",
  "representation": [{
    // used to contain a representation by value
    "id": "did:example:123456789abcdefghi#rep-1",
```

```
    "media-type": "application/json",
    "value": "some-json-structure-here"
  },
  {
    // used to point to a representation by reference
    "id": "did:example:123456789abcdefghi#rep-2",
    "seeAlso": "http://example.com/example"
  }]
}
```

Both options use the `id` subproperty to uniquely identify an instance of the `representation` property within the DID document using a DID fragment. This enables a unique DID URL to identify a representation of the resource.

The first option is to use the `value` subproperty to map a DID URL directly to a representation *embedded in the DID document itself*. From an RDF graph perspective, this corresponds to the following two RDF statements—the first one asserting the DID URL and the second one asserting the `rdf:value` of the representation:[5]

```
<DID>  <did:representation>  <DID URL>
<DID URL>  <rdf:value>  ...value...
```

While the DID identifies a non-information resource, the DID URL identifies an information resource **bound to the non-information resource via the DID document**. The DID *resolves* to a *description* of the non-information resource—the DID document. The DID URL *dereferences* to a *representation* of the information resource—the value of the `value` property directly contained inside the DID document. This is shown in Figure 3 (using JSON):

[5] See https://www.w3.org/TR/rdf-schema/#ch_value

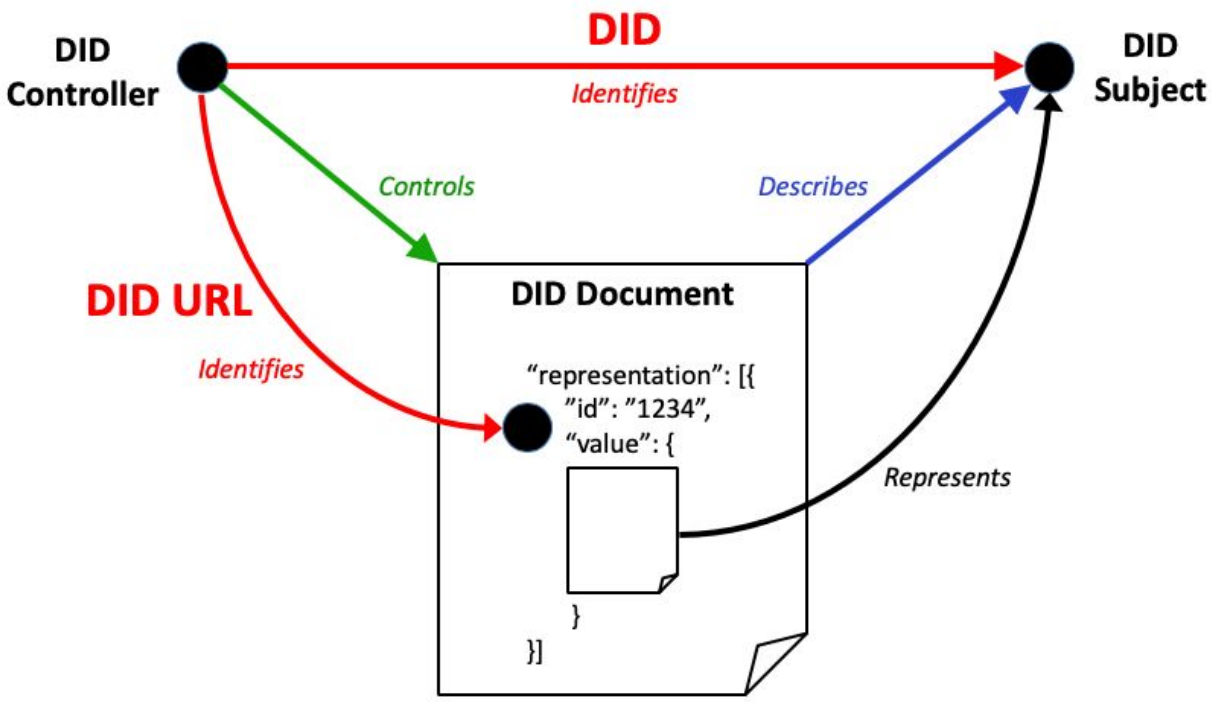

Figure 3: A DID document containing a representation of the DID subject as an information resource

The second option is to use the `seeAlso` property under the `representation` property to map a DID URL to a URI identifying a representation of the information resource external to the DID document. From an RDF graph perspective, this corresponds to the following two RDF statements—the first one asserting the DID URL and the second one asserting that the DID URL maps to another URI using the `rdfs:seeAlso` property:[6]

```
<DID>      <did:representation> <DID URL>
<DID URL>  <rdfs:seeAlso>             <URI>
```

In this case, the DID identifies the non-information resource, and both the DID URL and the `seeAlso` URI identify information resources **bound to the non-information resource via the DID document** . As always, the DID dereferences to a *description* of the non-information resource—the DID document. The DID URL dereferences to the value of the `seeAlso` property, which must be a URI. This effectively serves as a "logical redirect" to the URI for the information resource which in turn dereferences to a representation as shown in Figure 4:

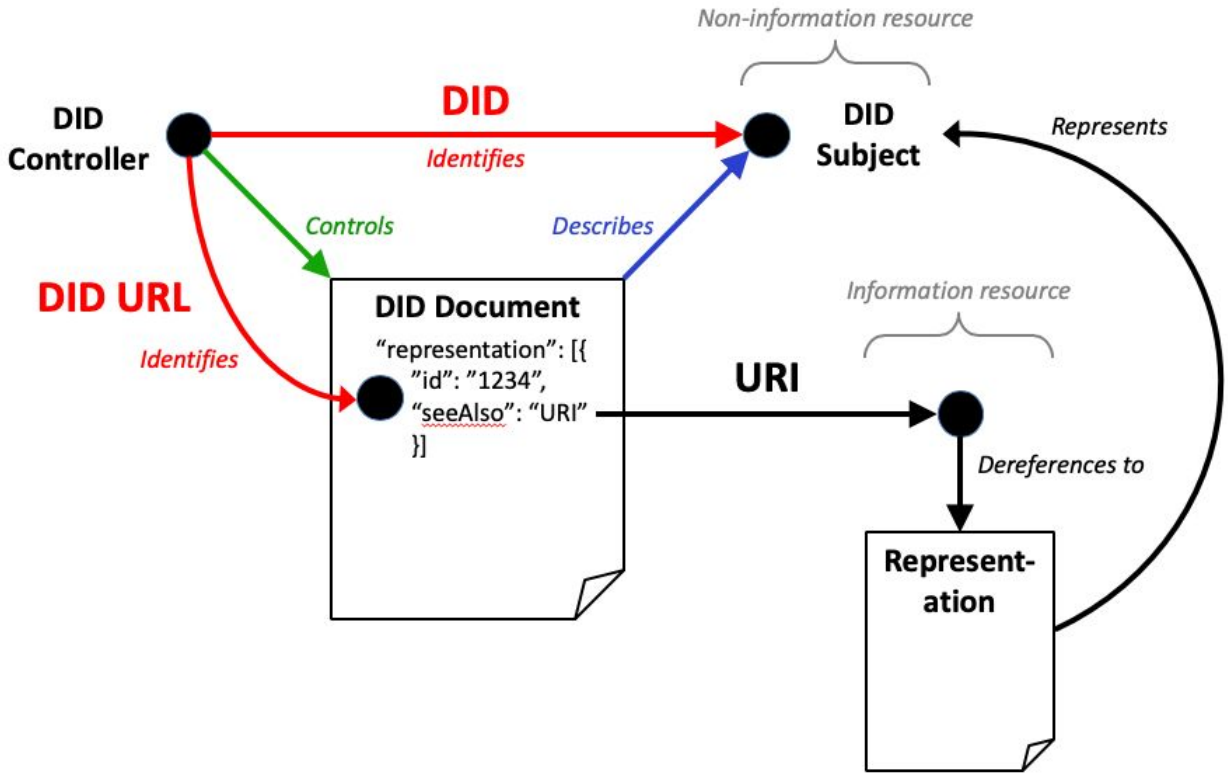

Figure 4: A DID document referencing a external representation of the DID subject as an information resource

[6] See https://www.w3.org/TR/rdf-schema/#ch_seealso

Note that the values of both the `seeOther` property in Figure 2 and the `seeAlso` property in Figure 4 are URIs. The difference is the meaning of these URIs from a Semantic Web standpoint:

- With `seeOther`, the URI is **an identifier for another information resource that describes the non-information resource** abstractly identified by the DID.
- With `seeAlso`, the URI is **an identifier for an information resource** abstractly identified by the DID.

It follows that a DID document should only include a `representation` property if the DID abstractly identifies an information resource, and it should only include a `seeOther` property if the DID abstractly identifies a non-information resource, *but never both*. If a DID document contains *neither* property is it indeterminate whether the DID abstractly identifies an information resource or a non-information resource.

Example use cases

Here are two examples to illustrate the real-world value of being able to unambiguously distinguish between information and non-information resources.

The first example is a web page about an author. This web page is identified with a standard HTTP URL. From the Semantic Web standpoint, this URL identifies exactly one information resource—a page of a website. The URL can be dereferenced to retrieve a representation of that resource—the current version of the web page. No dilemma here.

Now, if the same web home page is identified with a DID, it adds a layer of indirection. This is the whole purpose of abstract identification—the DID *abstractly* identifies the web page. As described above, it is easy for the associated DID document to reference the current URL for the web page—concrete identification—using the `seeAlso` property. The abstract DID is bound via the DID document to the concrete URL.

One benefit of this layer of abstraction is that *the DID never needs to change* even if the URL for the web page changes. DIDs effectively function as URNs (Uniform Resource Names)—persistent identifiers for information resources whose network location can change over time.[7]

Now let's take a second example—this time of a non-information resource: the author of the web page. Let's say she wants to create a URI to identify herself as the author in an RDF document describing the website. If the author used the URL of her web page, she runs into the Semantic Web dilemma: does the URL identify the web page as an information resource or the author as a non-information resource?

[7] https://tools.ietf.org/html/rfc8141

However if the author creates a DID as an abstract identifier for herself as a non-information resource, that DID will not be confused with either the URL for the web page or the DID for the web page. As explained above, the abstract DID for the author can resolve to a DID document that can reference the concrete URL for the author's web page (an information resource) using the `seeOther` property.

The result is a clean separation: a DID that abstractly identifies—and a DID document that describes—the author as a non-information resource, and a URL that concretely identifies a web page as an associated information resource also describing the author. Even better, the concrete URL for the web page can be discovered from the abstract DID. (If helpful, the author's DID can also be discovered from the web page if the author publishes it there—although for security reasons this DID should be dereferenced to verify that its DID document also a `seeOther` with the URL of the web page.)

This separation becomes even stronger if the author also has a DID for her web page. Now the author can use the first DID to identify herself as a person and the second DID to identify her web page. Both of these DIDs are permanent, semantically distinct, cryptographically verifiable, and under her personal control for as long as she wants them.

Appendix B: DID Controllers and DID Subjects

The relationship between DID controllers and DID subjects can be confusing. The W3C DID Working Group has found it helpful to classify DID subjects into two disjoint sets based on their relationship to the DID controller.

Set 1: The DID subject is the DID controller

The first set, shown in Figure 1, is the common scenario where the DID subject is also the DID controller. This is the case when an individual or organization creates a DID to self-identify.

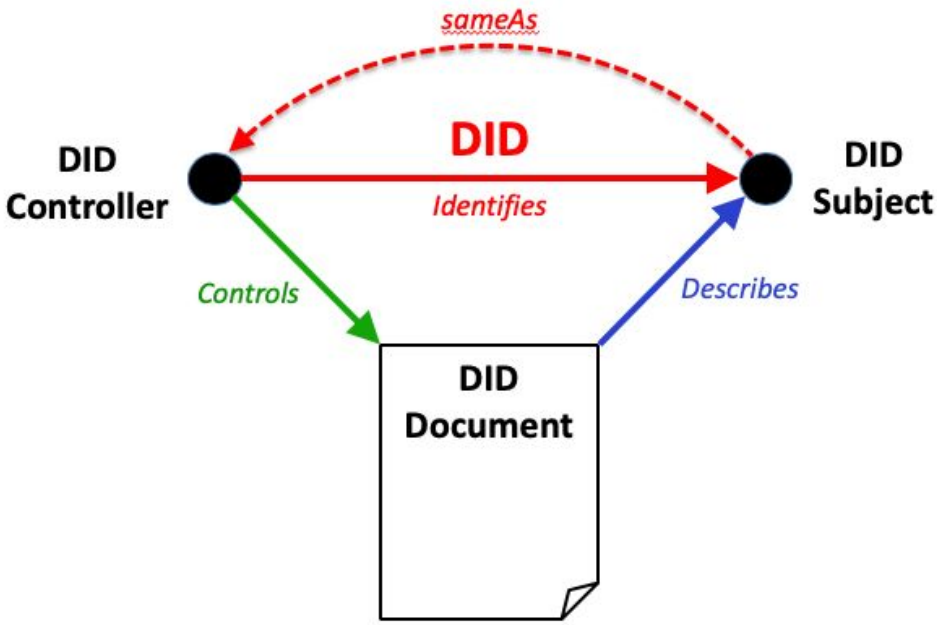

Figure 1: The DID subject is the same entity as the DID controller

From a graph model perspective, even though the nodes identified as the DID controller and DID subject in Figure 1 are distinct, there is a logical arc connecting them to express a semantic equivalence relationship (in RDF/OWL, this is expressed using the `owl:sameAs` predicate).

Set 2: The DID subject is not the DID controller

In the second case, shown in Figure 2, the DID subject is a separate entity from the DID controller. This would be the case when, for example, a parent creates a DID for a child; a

corporation creates a DID for a subsidiary; or a manufacturer creates a DID for a product, an IoT device, or a digital file.

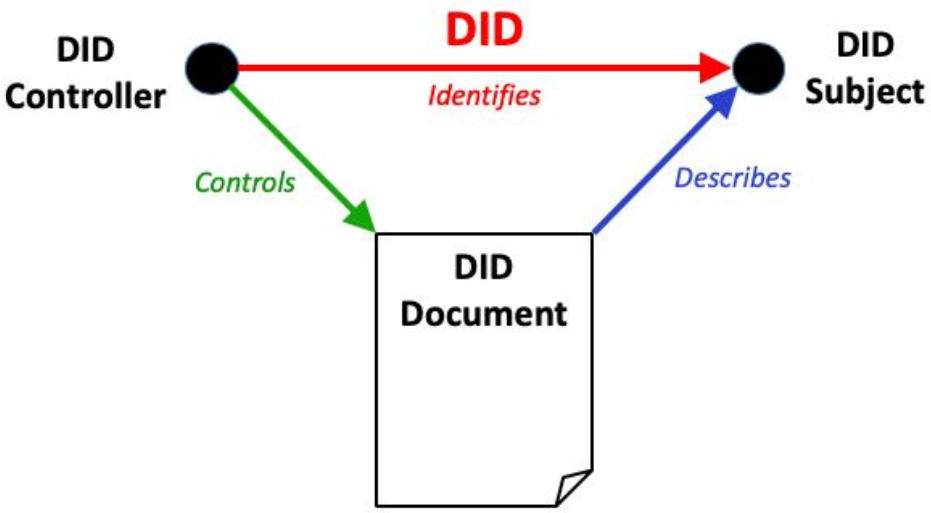

Figure 2: The DID subject is a separate entity from the DID controller

From a graph model perspective, the only difference between Figure 1 and Figure 2 is that in the latter there is no `owl:sameAs` arc connecting the DID subject and DID controller nodes.

Also note that, as explained in Appendix A, the DID document is always a *description* and never a *representation* of the DID subject, no matter whether the DID subject is or is not the DID controller.

Appendix C: Multiple DID Controllers

In both cases described in Appendix B, a DID document may have more than one DID controller. In this situation there are three logical options available to the DID controllers.

Option #1: Independent DID Controllers

In the first option, all the DID controllers may all act separately, i.e., each of them has full power to update the DID document. In this configuration (shown in Figure 1):

- Each additional DID controller is another distinct graph node.
- The same arc (“controls”) exists between each DID controller and the DID document.

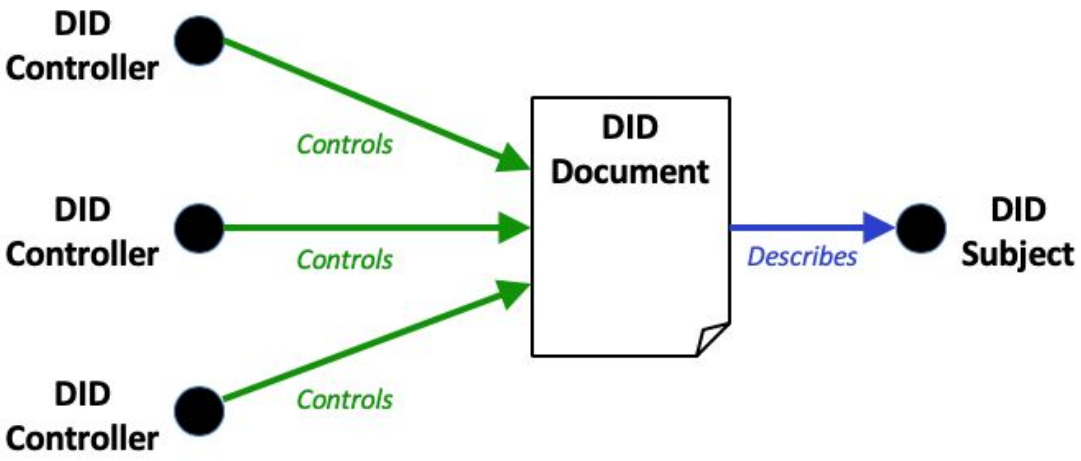

Figure 1: Multiple independent DID controllers who can each act independently

Option #2: Aggregate DID Controllers

In this option, all of the DID controllers must act together, such as when using a cryptographic multisig algorithm. This case is functionally identical to a single DID controller as all the DID controller nodes collapse into the DID controller node as shown in Figure 2:

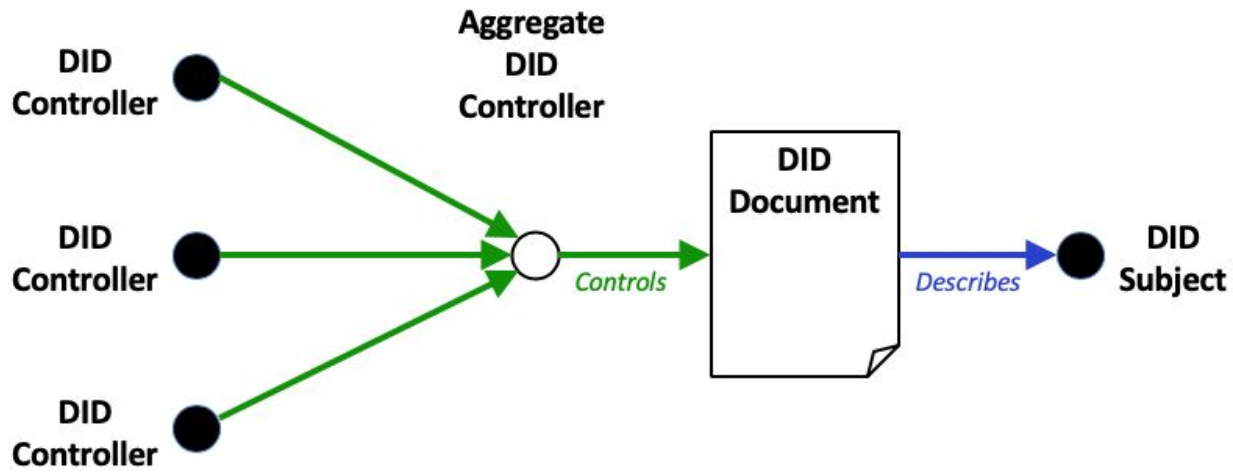

Figure 2: Multiple DID controllers who must all act together as a single aggregate DID controller

Option #3: Partial Aggregate DID Controllers

In this option, some subset of the DID controllers must act together, such as when using an m-of-n cryptographic signature algorithm. This is a variant of option two where only a subset of the DID controller nodes are needed to collapse into the DID controller node. This is shown as dotted “control” arcs in Figure 3:

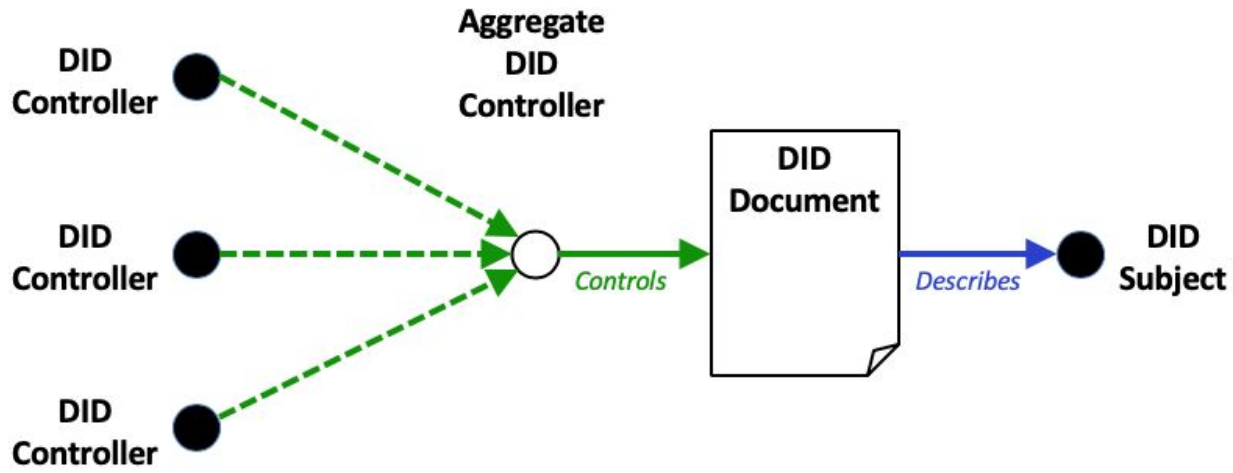

Figure 3: Multiple DID controllers who must act in some combination as a single DID controller

Notes:

1. These DID controller options can be further nested in any combination.
2. In all three of these configurations, **only one DID controller** may be the target of an

RDF/OWL `sameAs` arc from the DID subject as shown in Figure 1 of Appendix B.