This is a SECOND REVISED proposal for the W3C DID Core Specification. The original proposal (now deprecated) is here. The first revision (also deprecated) is here. Please direct all comments to the DID Core issue thread.

Appendix A: What does a DID identify?

The DID Core 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. However some applications of DIDs, particularly those involving the Semantic Web, may need a more precise definition of how DID identification works. That is the purpose of this Appendix.

What can a DID identify?

Since a DID is a specific type of URI, the answer to this question is provided by section 1.1 of the URI specification (RFC 3986):

> *This specification does not limit the scope of what might be a resource; rather, the term "resource" is used in a general sense for whatever might be identified by a URI. Familiar examples include an electronic document, an image, a source of information with a consistent purpose (e.g., "today's weather report for Los Angeles"), a service (e.g., an HTTP-to-SMS gateway), and a collection of other resources. A resource is not necessarily accessible via the Internet; e.g., human beings, corporations, and bound books in a library can also be resources. Likewise, abstract concepts can be resources, such as the operators and operands of a mathematical equation, the types of a relationship (e.g., "parent" or "employee"), or numeric values (e.g., zero, one, and infinity).*

It does not matter whether a resource is “on” or “off” the Internet—if it can be identified, it can be assigned a URI, and therefore it can be assigned a DID.

How do you know what a DID identifies?

For any DID, the DID controller determines the DID subject. Unfortunately it is all but impossible to determine this from looking at the DID itself. The reason is that in order to satisfy several core properties of a DID as an identifier—persistence, decentralization, and cryptographic verifiability—DIDs are generally only meaningful to machines, not humans. To illustrate, compare the following two URIs:

```
https://www.w3.org/2019/did-wg/WorkMode/getting-started

did:example:8uQhQMGzWxR8vw5P3UWH1j
```

The first is the URL of the Getting Started page of the W3C DID Working Group. This is a human-meaningful identifier (to someone who understands the English language). In this sense, the reader can be said to “know” what the URL identifies without having to dereference it.

The second—the example DID—is meaningless to humans no matter what language you speak. What it identifies is anyone’s guess in the absence of further information describing the DID subject. So further information about the DID subject is only discoverable by resolving the DID to the DID document (or via some other description of the DID).

Does the DID identify the DID document?

No. To be very precise, the DID identifies the DID subject and *resolves to* the DID document (by following the protocol specified by the DID method). In other words, the DID document is an artifact of DID resolution that describes the DID subject but is not a separate resource by itself. This distinction is illustrated by the graph model shown in Figure A1.1.

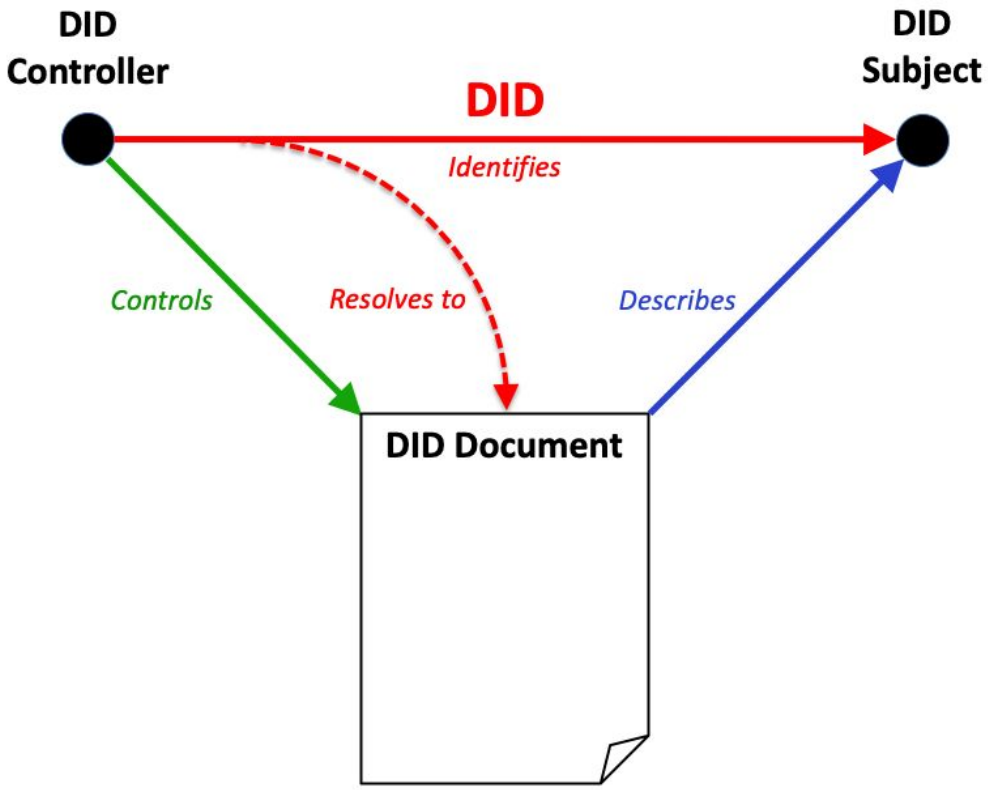

Figure A1.1: As an identifier, a DID identifies a DID subject and resolves to a DID document that describes the DID subject. The DID document is an artifact of DID resolution.

How does the DID document describe the DID subject?

Each normative property in a DID document is a statement by the DID controller describing the DID subject. Because there is only one required property in a DID document—the `id` property—the only statement guaranteed to be in a DID document is the one shown by the solid red arrow in Figure A1.1 asserting that the identifier of the DID subject is the DID.

Every other normative property in the DID document describes an attribute of the DID subject. For example:

- The `type` property describes the nature of the DID subject (person, organization, book, web page, data structure, abstract concept, etc.)
- The `alsoKnownAs` property describes other URIs that identify the DID subject (see the next section).
- [TODO—list additional examples]

How can you discover more information about the DID subject?

This is the purpose of the `alsoKnownAs` property. The DID controller can use it to provide a list of other URIs (including DIDs) that identify the same DID subject. Resolving or dereferencing

these URIs may yield other descriptions or representations of the DID subject as illustrated in Figure A1.2.

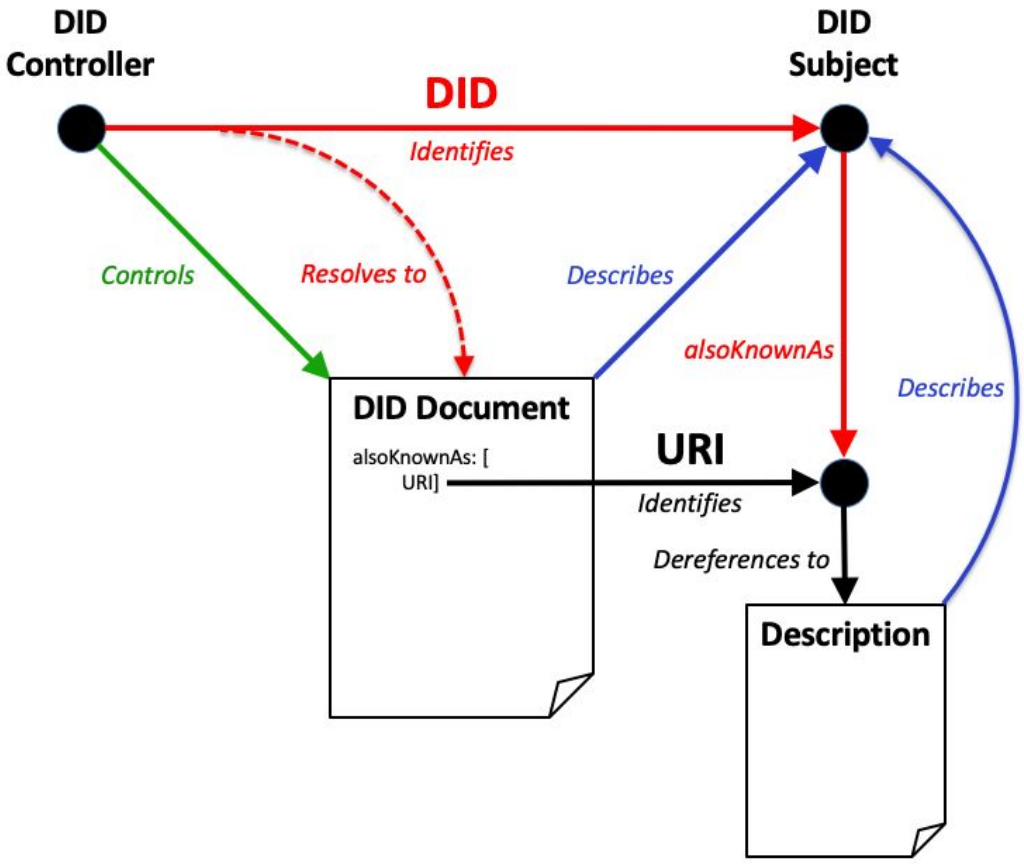

Figure A1.2: A DID document can use the `alsoKnownAs` property to assert another URI (including another DID) that identifies the same DID subject

This mechanism is how DID identification fulfills a longstanding recommendation from the W3C:[1]

> *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.*

[1] "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 can use that format to meet the letter of this W3C recommendation.

Can the DID document serve as a representation of the DID subject?

If the DID subject is an information resource that can be retrieved from the Internet, then yes, the DID document can serve as a representation of the DID subject. For example, a data schema that needs a persistent, cryptographically verifiable identifier could be assigned a DID, and its DID document could be used as a standard way to retrieve a representation of that schema.

Semantically, it is recommended to identify this type of resource using the `type` property of the DID document. For example, the following statement in a DID document would assert (in JSON) that the DID subject is an invoice.

```
"type": ["https://schema.org/Invoice"]
```

The `type` property can be used to avoid any ambiguity regarding whether a DID identifies an information resource retrievable from the Internet (such as a web page) or a non-information resource that is not retrievable from the Internet (such as the author of that web page).

For example, say the author of a book wants to create a page on the web describing herself so readers of the book can go to that web page to learn more about the author. The author also wants to create an RDF document with a machine-readable description of the book that identifies her as the author. As the W3C points out,[2] it would be semantically confusing if that RDF document used the URL for the web page describing the author as an identifier for the author as a person.

The author can easily solve that problem by creating a DID for herself and placing the following type statement in the DID document:

```
"type": ["https://schema.org/person"]
```

Now the author can use the DID to identify herself in all of her publications without an ambiguity about what that DID identifies.

Even better, the author *could* also create a DID for the web page. In that DID document, she could place the following statements:

```
"type": ["https://schema.org/WebPage"]
"alsoKnownAs": ["https://example.org/myhomepage/"]
```

Even though the web page already has a URL, the DID adds a layer of indirection. It 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

---

[2] "Cool URIs for the Semantic Web": https://www.w3.org/TR/cooluris/.

location can change over time.[3]

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

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 case, shown in Figure A2.1, 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.

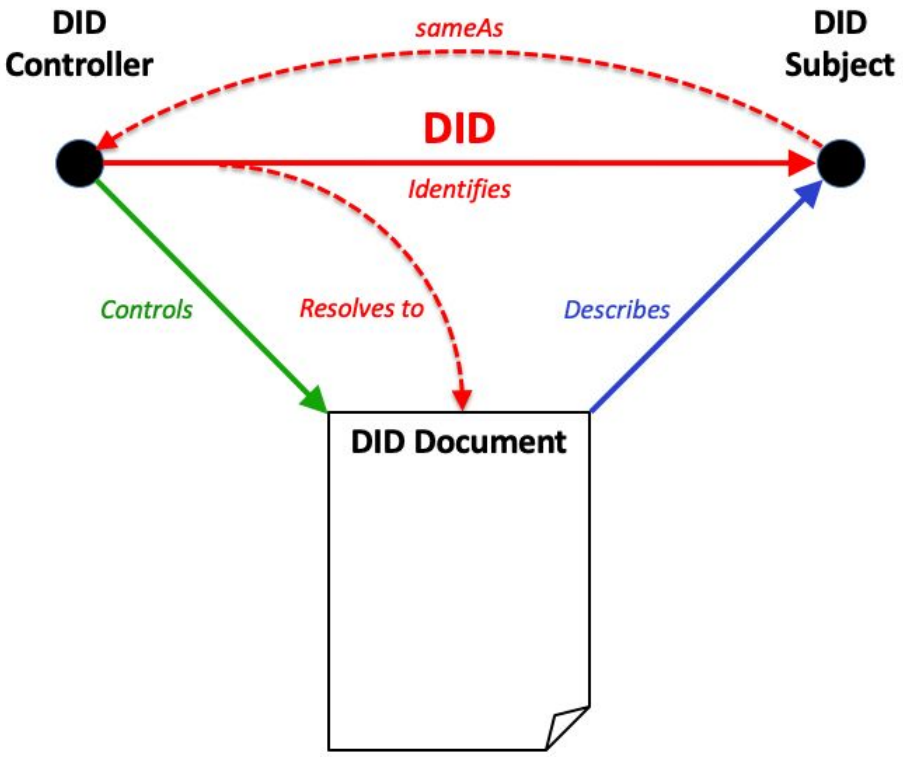

Figure A2.1: The DID subject is the same entity as the DID controller

From a graph model perspective, even though the nodes identified as the DID controller and DID subject in Figure A2.1 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 <u>not</u> the DID controller

The second case, shown in Figure A1.1 of Appendix A, is when 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.

From a graph model perspective, the only difference between Figure A2.1 and A1.1 is that in the latter there is no `owl:sameAs` arc connecting the DID subject and DID controller nodes.

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.

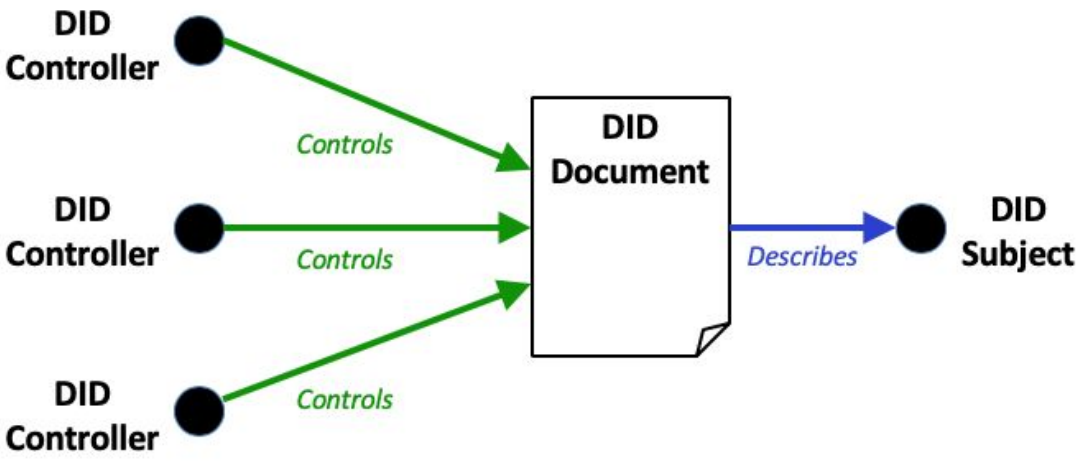

Figure A3.1: Multiple independent DID controllers who can each act independently

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:

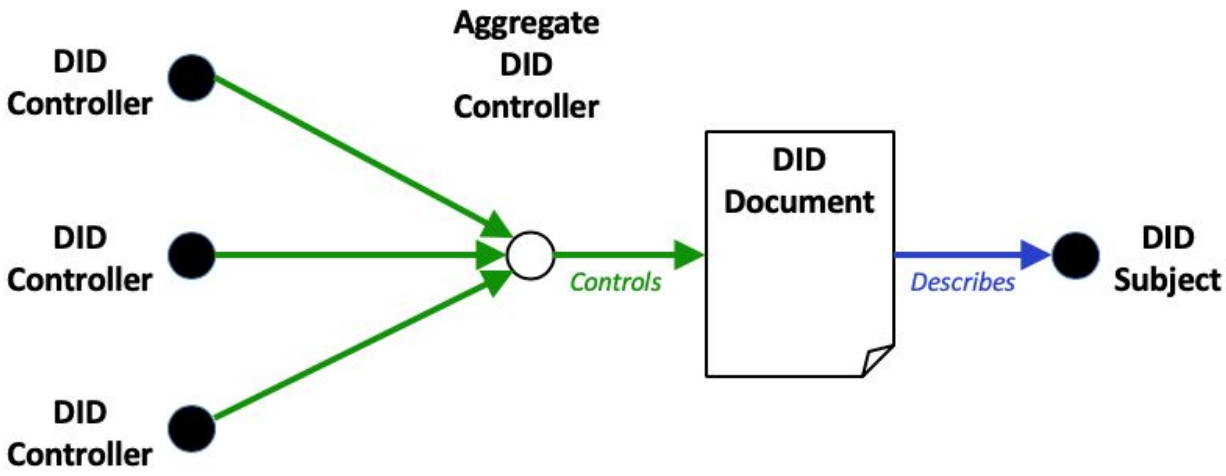

Figure A3.2: Multiple DID controllers who must all act together as a single aggregate DID controller

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:

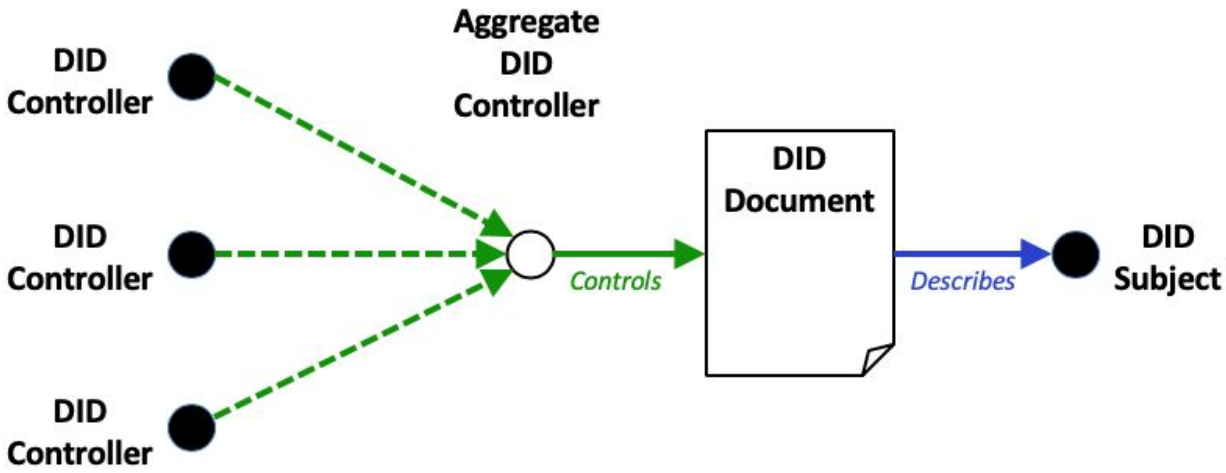

Figure A3.3: Multiple DID controllers who must act in some combination as a single DID controller

These DID controller options can be further nested in any combination. However, note that 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 A2.1 of Appendix B.