Dylan W. Sheehan

CS 4000-013: Undergraduate Research

Decentralized Identifier (DID)

Computer Science Major

Dec. 2022 - Jan. 2023

TABLE OF CONTENTS

1- Introduction

Throughout the digital age, we have used many solutions to identify ourselves and developed new types of technology to create easier methods for identification in order to complete transactions, access private information such as banking information, and even use private services from different vendors. According to Trulioo (RegTech company building and connecting digital identity frameworks) infographic history document [Reference 1], The first type of identification that became extremely prominent was biometrics. Biometrics was implemented in the United States in 2004 as a way to catch criminals and offenders and soon this idea spread to different breakthroughs in technology such as using fingerprints, speech recognition, DNA sequencing, hand geometry, vascular pattern recognition, and much more. Soon biometrics will start to have taken their place in the form of a digital ID system where they will have been implemented in hardware such as phones, computers, and other devices to make sure the user is authenticated to access private information. There are many commercial applications like Amazon that have taken their own step and created many different databases to hold user data for their own services such as their eCommerce site, their brands and products, and also information relating to essentials in their home and for delivery. All of this information regarding addresses, payment methods, and descriptions or locations of certain products are all stored in their sanctioned directories which are usually hidden behind a user's biometrics or digital records. As advanced biometrics grow, we start to see more issues being developed. There are many types of fraudulent practices with digital identities being manipulated, identity fraud being committed, and money laundering activities happening almost daily. There have been many suggestions to tackle this type of activity such as reducing the digital ID overhead and making identification simplistic so it can prevent misuse and fraud, or creating standardized identifiers so it will be harder to grab sensitive information but easier to retain public keys for users. There is also the issue of expanding identification systems in order to process different types of objects, animals, web services, users, versions of software, etc. so everything can be easily coordinated within a team or easily documented to make certain people's jobs easier. Some of these objectives are measured in global electronic verification where each specific service provides a way to identify different items that are fully dependent on their organization. One example of this is seen in a standard such as KYC or "Know Your Customer" where the organizations develop ways to identify individuals on how they use their products and establish customer identity. However, most of these solutions are proven to be very time-consuming with too many variables to take into account along with having the need to provide each separate active directory with ways they can tell who a customer or user is along with many different authentication methods that can vary between other organizations.

As you can infer, digital identifiers are very important in the digital world because they are becoming ubiquitous and we need a way in order to evolve identity with this changing world. More and more items such as books, people, retail items, companies, institutions, genes/proteins/viruses, websites, software versions, commits, vehicles, planes, internet devices, IoT (Internet of Things), and even abstract concepts are becoming very complicated and low-leveled that they need some type of identification that can be accessed to understand its theoretical and conceptual nature and used to compare it with something that is similar. For instance, a person patching a software version to fix a specific bug in a music player app will be vastly different from another person patching the version since it can be done in different ways. For ease of use and reading in a timely manner, a company can read their identifiers to generate an official version. Of course this is one example, this can be more simple such as an ISBN of a book. What if in the future, we have digital produced articles that throughout time become centralized books. What if guides have become expanded to include different ways on how to do something? As you can see, a simple ISBN number can't identify these types of scenarios but a certain decentralized system can. Current identifiers nowadays are not "persistent" which means that if something about a user for example, is not used anymore, then it disappears forever and cannot retain that knowledge. This can be important for marketing or sales workers who can identify specific tactics for their business that have not worked or have worked in the past.

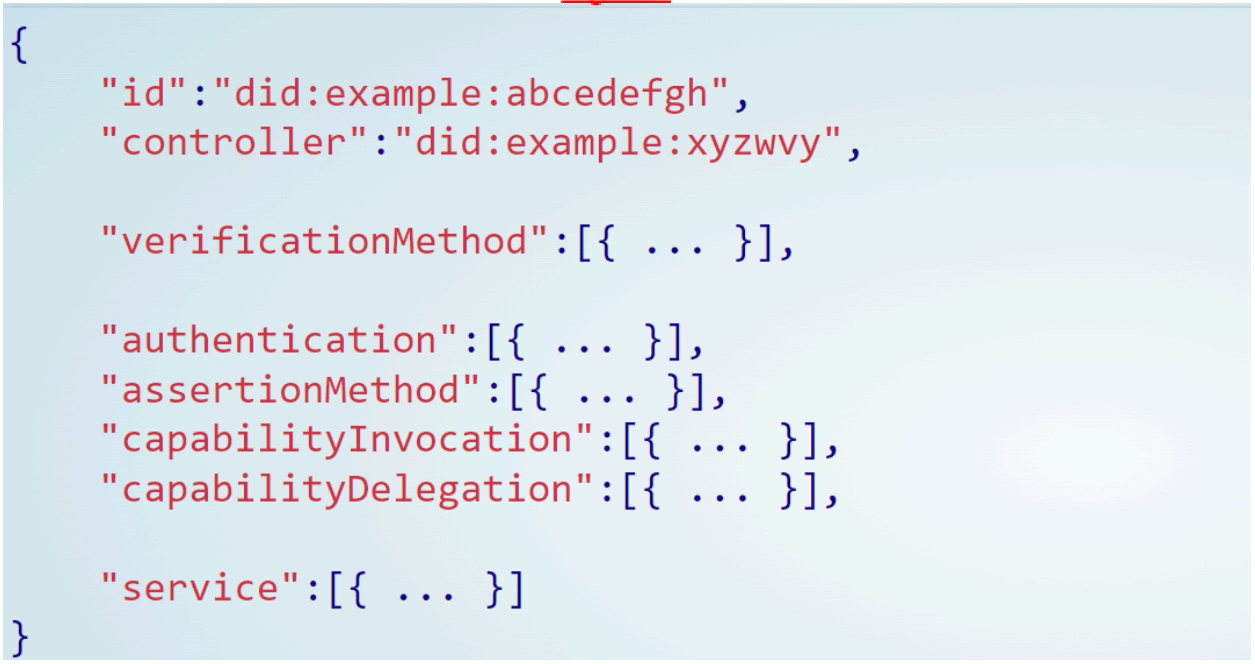
As identity is still evolving, one particular type of identification is emerging to combat the need of a changing identification system, fraud, and facilitate current systems for businesses, commercial applications, or web services. According to the World Wide Web Consortium (W3C) inception [Reference 2], Decentralized Identifiers or commonly referred to as DIDs are a new type of identifier on the rise that enables a completely verifiable, decentralized identity. As discussed, a DID can be any subject, hence it is "abstract" which means that it identifies a person, an entire organization, a "thing", a data model, an entity, etc. These DIDs are controlled by a DID controller who is the one that can interact and edit the information for a specific DID in a DID Document in which is decentralized and stored without any type of coordinated or specified database or ledger (discussed later). DIDs are great because they are designed to be decoupled from centralized registries, identity providers, and certificate authorities. This means in general, DIDs should follow a set of goals to make identification easier (W3C, Section 1.2). The first goal is 'Ease of Creation' which means that it should be quick and cheap to create possibly thousands of DIDs. This is one of the issues that identification has trouble solving. There are many identifications which take way too long to process and are not cheap to produce. The next goal is the most important: "Decentralized". As I mentioned earlier, decentralization was one the main solutions to a lot of identity management in which companies are slowly gravitating to. It prevents the need of hassle to have these centralized registers for every single product they sell and every single customer that is signed up or registered for their services. A supermarket chain can simply incorporate DIDs for the food and utilities they sell along with using these DIDs to incorporate with the DIDs of the customer that also has more identification and authentication protocols written on their DID documents. This rids the unnecessary barcodes and rewards programs that many of these supermarkets incorporate which can be time consuming when they can simplify the processes. The next goal that I mentioned earlier was identification being “Persistent” which should permanently be tied to the subject it is referring to so it does not have to be vanished. This is useful for many fraudulent reasons and personal reasons if someone needs to refer to something that they had or did in the past. Such as a University Degree. A college degree can have its own DID to refer to what the degree is and where it is from and who has received it. Another goal is “Resolvability” which should be able to quickly identify all the information about a subject of the DID. This can be very helpful with medical personnel or in the medical space. If someone faints or collapses, the medical personnel can pull up one’s DID that contains their blood type, allergies, or vaccination details, etc. for quick treatment. Although a company like Apple is trying to incorporate their own software to be used within their recent iPhone models, one still has to verify they are medical personnel and have the right to view their medical information without breaching confidentiality. However, a DID can be automatically provided by the victim if something goes wrong since you can incorporate more information in it such as authentication and authorization purposes for specific medical providers (more about authentication later). The last and most important goal of a DID is that it should be “Cryptographically verifiable” to prevent identity theft. Due to DIDs being decentralized from federated identifiers or some type of centralized registry, we need a way to prove the identity and ownership of something using cryptography through some type of mechanism. URIs are used to “bridge” the DIDs to the DID Document in order to provide trustable interactions for the specific subject that is using the DID. Also recall that a DID is abstract, this means that its documentation can be versatile for the subject by its controller and can include many different types of cryptographic data that can be used for key agreements, authentication, assertions (credentials), and invocation and delegations to use a certain API. As you can see through this brief introduction, Decentralized Identifiers can be used anywhere and is a better way for software and hardware developers to incorporate specifications for further use for an organization, business, or communication between individuals or other systems and services. Throughout this report, I will be discussing what a DID is more in depth along with its core properties, the benefits and some problems with DIDs, some interesting features of a few DID software products, along with some challenges that DIDs should be researched in the future.

2 - What is a Decentralized Identifier?

Throughout most of the introduction, I have discussed the basic idea of what a Decentralized Identifier (DID) is, what are the main goals this new identification idea is supposed to achieve, and how it is an improvement on the other types of identification systems. Although approaching the concept of DIDs at a high level can give a great overview and the possibilities they can grant, it is much better explained in depth with examples and a conceptual understanding of the DID architecture (component model), DID identifiers (in-depth syntax) and DID Data Models, and the DID core properties and representations such as verification methods and relationships for daily use so we can conclude its future use and the types of identification systems that it can possibly replace. Recall that a DID is a globally unique identifier that is designed to enable individuals or organizations to generate identifiers using the systems they trust through entities which prove control by authentication methods such as digital signatures or other cryptographic proofs. On top of the goals mentioned in the introduction, DIDs still need to support control, privacy, security, and interoperability so they can be used as components of larger systems. This is to ensure that the users can trust these new systems and that the data being used can be reliability safe and not cause issues of concern. Control and security is needed to give human and non-humans the power to control their digital identifiers and provide significant security protocols without having the need to rely on a specific external authority. Privacy is also a big deal when it comes to identification. Similar to being cryptographically verifiable, the users data needs also to be private and to only be disclosed if granted permission upon. Interoperability is also important for incorporating certain software libraries and tools into a system so DIDs can use these dynamically for other entities. Most of the entities that are defined in a DID document can contain thousands of lines of data related to a specific DID and can be explained through a simple example. According to W3C [Reference 2], a DID is described as a "simple text string consisting of three parts: the URI scheme identifier, the identifier for the DID method, and the DID-specific identifier" in which all of these resolve to the DID document (W3C, Section 1.1). Before discussing the DID document, let's break down the meaning behind what the DID stands for. The URI scheme controller for DIDs will simply be written as "did" to indicate that URI is a DID identifier. According to the Library of Congress Standards [Reference 3], URIs can be anything related from a URN such as a name in a namespace, a URL such as "https", or even an information asset like "info". Because the "did" scheme is a global identifier and can be linked to associated DID documents for information, it can be used in the IETF (Internet Engineering Task

Force) or even the world wide web (Library of Congress, Section 4). For instance, in web development, we can use DIDs in HTML tagging such as tagging it into an anchor tag with its 'href' attribute pointing to a "did" scheme instead of an "http" scheme when working with forms or other types of elements. Even better, there are many tools out there that help manage and break down DIDs so they are easier to understand and comprehend. The second part of the DID is called the DID method in which it works with a method scheme that gives the idea of what the DID is for and what it is supposed to do. The DID method follows the DID scheme with a colon ":" to indicate that it is the start of the next text string. The last part that follows with another colon is the DID method-specific identifier in which it is usually describing what the method is doing and its purpose. According to W3C [Reference 2] and the FETP (WSC, Section 1.1), usually this number is generated either to indicate a reference or some other specific reason that the method was used. Note that the identifier is entirely dependent on the DIDs goal or what it is specifically identifying such as a transaction. In the upcoming years, many software engineers are slowly developing a convergence or methods into singular methods to meet the DIDs goal of 'Ease of Use' as mentioned in the introduction to this report. If we put everything together, we form a DID such as: 'did:btcr:xyv2-q9wap7t' which tells us that this is a DID built on top of the Bitcoin blockchain and the method-specific identifier after the second colon tells us that it represents a transaction that occurred. This DID like all other DIDs refers to its DID document which is tightly coupled and gives specific information in which is needed for reference or for information. All DID documents are represented differently. According to Ivan Herman from W3C, DIDs are also commonly referred to as 'DID+DID Document' since the documentation is super important to include the right features and its speciality in incorporating cryptographic data to be used as a keychain for various cryptographic applications. All DID Documents can be serialized such as a JSON, JSON-LD, or a CBOR file and have plenty of attributes that may describe the DID in one form or the other. A typical DID document structure can be described in *Figure 1* below which is an example JSON file provided from W3C [Reference 2]. Notice that each DID document contains its 'id' DID as shown in the figure which tells the link between the document and its DID. However, notice that there is a controller DID which is always included to tell who is in charge of the DID. This DID can represent a person or any subject that controls

Figure 1

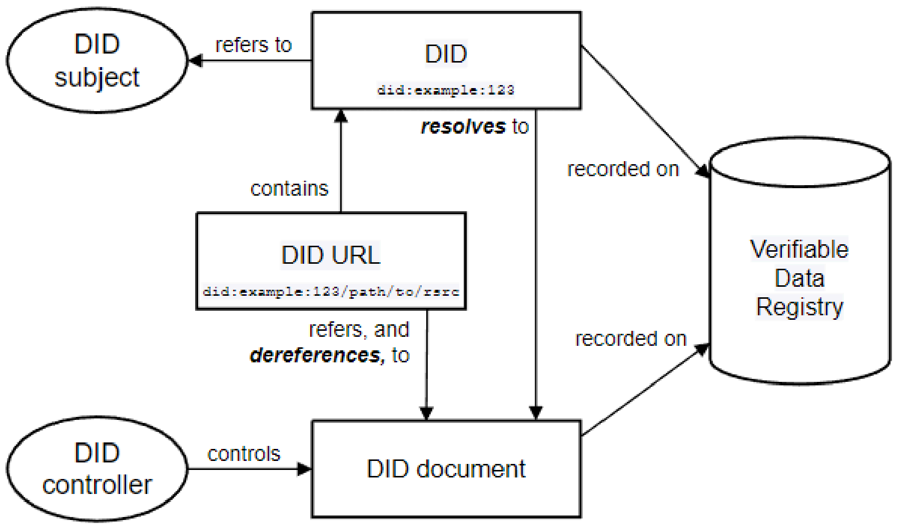


```
{
    "id":"did:example:abcedefgh",
    "controller":"did:example:xyzwvy",

    "verificationMethod":[{ ... }],

    "authentication":[{ ... }],
    "assertionMethod":[{ ... }],
    "capabilityInvocation":[{ ... }],
    "capabilityDelegation":[{ ... }],

    "service":[{ ... }]
}
```

the information that is needed and stored in the DID. Also there are a variety of verification methods as suggested by the "verificationMethod" tag. These verification methods can be used to include various public keys that are associated with the DID or even can include DID authentication usages themselves. Recall that each public key usually includes some sort of 'type' attribute and the public key itself that is usually encrypted. This works similarly to the "authentication" attribute in which contains keys that are used to verify the controller or it can refer to the keys listed separately in the verification methods of the controller so it can be used for verifying certain holders. This can also include full keys that can be used for authentication only or 1 specific key in order to access the specific DID. Lastly, service endpoints usually indicate that the DID involves some type of service or a branching service that it depends on to operate. This can be related to advertising for instance where the specific business has an application where it contains specific services to advertise to the user when interacted with. Before diving into an in-depth analysis of these attributes in the rest of this section, I am going to give a brief overview of the architectural design that explains the overall process of how a DID works and how components such as the documentation fit into play.

The Architectural overview of a Decentralized Identifier is one of the easiest ways to summarize how it works and what components come into play to make it understandable. The best way to explain the DID is similar to that of a cryptocurrency. If we take a specific digital token such as Bitcoin, and we look at the structure of it, we see how it is decentralized and we have specific ledgers that constantly record information to prevent fraud. Each coin is stored on the blockchain and has generated hash values to verify transactions. A DID works similarly when verifying identity. Below in *Figure 2* from W3C [Reference 2] portrays how the DID works in correspondence to its document it produces along with how the subject is referenced (W3C, Section 1.3). Notice that each part of the architecture can be used combined as one component for a hardware or software system as I mentioned previously. From the architecture,

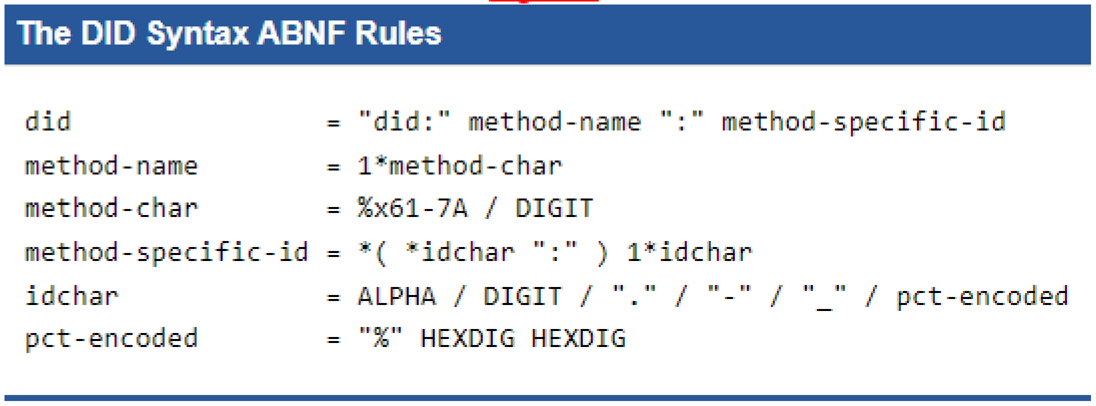
Figure 2



we can see that the DID is mentioned at the very top and it refers to a subject which is always a person, group, or concept that the DID wants to reference. The DID URL is the extra information so we can find specific parts from the DID that the URL references or dereferences in the documentation. This is given from the DID Method discussed previously in which this part will give us information on how a DID is referenced or dereferenced. This can be a path, query, or fragment of some sort that we need to access. The DID controller as mentioned earlier, is how we control the DID and make changes to the documentation. The DID controller does not have to be always controlling the subject. The subject can be the controller itself if it is a private DID or if the subject needs multiple controllers, it can be the form of a multi-owned controller. Lastly, the most important piece is our DID document and how it is recorded on the data registry. Recall earlier I mentioned Bitcoin and how each transaction is usually recorded on distributed ledgers where we can verify transactions on billions of computers. According to Seth Shobhit on Investopedia [Reference 4] which describes how public ledgers in cryptocurrency operate by consisting of a large network with each node, also called 'full nodes', to maintain a copy of each verification of transaction. Similarly, if we think about verification data registry, it can include this idea of distributed ledgers because it has worked before with cryptocurrency except for storing each verifiable transaction we store the DID documents with each public key associated to a specific DID. According to W3C [Reference 2], we can also use the idea of "coexisting" DIDs "on top" of the bitcoin blockchain to provide an extra layer of security transactions due to the fact that this idea is still new, there can be issues of more criminals seeking to find and crack these documents (W3C, Section 1.3). There are also other ways such as using decentralized file systems in which store records on computers or using databases which is also a very reliable idea since most of the use of digital wallets today are centered around databases for centralized systems such as Amazon Web Services. Of course, there are plenty more methods on how we can store DID documentation like peer-to-peer network exchanges, but it also highlights that there isn't necessarily one way to accomplish this, as long as it is decentralized and you don't need to provide extra documentation to prove an existence of an entity or even simple as something such as a transaction. The next thing to explore is the syntax and how the DID identifies itself with an understanding of how the DID is architecturally structured under the hood. Although I gave a general overview in some examples previously, it is important to understand how it grabs and represents information and ultimately how it works.

Decentralized Identifiers have unique syntax that uses the URI scheme and follows the ABNF (Augmented Backus-Naur Form) and each part of the syntax can be generalized in *Figure 3* given from W3C [Reference 2] (W3C, Section 3.1). Starting from 'did' and going down we

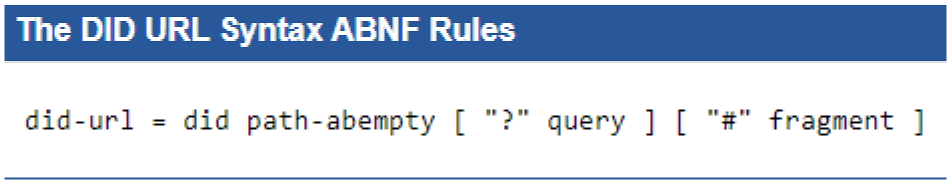
Figure 3


The DID Syntax ABNF Rules

```
did                = "did:" method-name ":" method-specific-id
method-name        = 1*method-char
method-char        = %x61-7A / DIGIT
method-specific-id = *( *idchar ":" ) 1*idchar
idchar             = ALPHA / DIGIT / "." / "-" / "_" / pct-encoded
pct-encoded        = "%" HEXDIG HEXDIG
```

can see there are different ways to express the DID but ultimately it follows the 'did: method-name: method-specific-id' format. There are plenty of ways to represent methods but for the simplicity of this report, I will briefly discuss it between sections. There is also the DID URL Syntax representation which W3C [Reference 2] calls a "network location identifier" which was discussed in the DID architecture previously. This identifier goes through the network to find a specific resource that is needed for the DID such as the DID subjects that it represents, the verification methods that are needed, and the services that are used such as social networking or file system services. The DID URL scheme is provided in *Figure 4* from W3C [Reference 2] below which behaves similarly to *Figure 3* due to the fact that both follow ABNF and build upon each other (W3C, Section 3.2). Remember that the DID URL acts like the middle path between

Figure 4

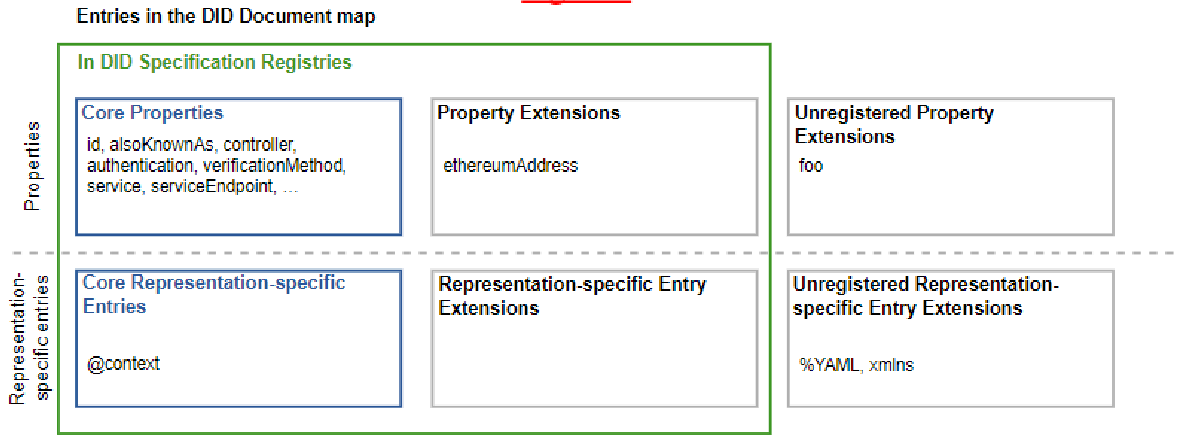


The DID URL Syntax ABNF Rules

```
did-url = did path-abempty [ "?" query ] [ "#" fragment ]
```

the DID and the documentation so we can find specific items that coordinate between what the controller is looking for about the subject(s). Notice that there are 3 identifiers: a path, a query, and a fragment with each of these components allowing developers of these systems to specify specific items either within documentation or what the 'did' represents itself. According to Tyronzil [Reference 5] which is actually a method identifier for a Decentralized Identifier and part of the Ziqilla blockchain, contains information in their documentation that states that the DID path is used as an address specified from the controller to coordinate to a certain endpoint (Tyronzil, Scheme Section). In *Figure 4,* it is represented by 'path-abempty' and always starts out after the 'did' operation. One example that is given in W3C is 'did:example:123456/path' which shows that we can use a path to signify a service. Tyronzil [Reference 5] also discusses the query component which is the 'thing' we need to grab from the document given the specific path. If the DID does not contain a path to resolve to a specific DID documentation query, then it is assumed we grab an element from the document straight from the DID method that it specified. Notice in *Figure 4* the query is given to us with a "?" token before so we know where to put the query similar to that of a normal URL in web development. W3C [Reference 2] gives us an example such as 'did:example:123456?versionId=1' which means we use the path of '123456' service endpoint to find the 'versionID' component in the documentation that resolves to 1. The DID fragments is a retrieval component that grabs resources, services, rules, or verification methods in the DID document. For example, W3C [Reference 2] gives us an example on how to retrieve a public key for a document given by ‘did:example:123#public-key-0’. Notice that for a fragment to work accordingly there needs to be a field by the specified query (if given) and that there should be a “#” token before the query is stated. Since the query can be given as almost anything since it acts like a folder for fragments to find the data within the DID document, we can use parameters that signify where or what resource is being requested. However, recall that each DID method similar to those in the DID URL requires their own generalized methods and identifiers to use as a representation for a document. In other words, each identifier method has their own parameters specifically to them. For instance, Tyronzil [Reference 5] is part of the Tyron Self-Sovereign Identity Protocol and they have specific parameters for each of their queries. However, each of these identifier methods have a common basis of five parameters that can describe what the controller is looking for in the DID document. The first parameter is service in which it identifies a service from the DID document by the service ID. For instance, if we were using Tyronzil [Reference 5] DID URL:‘did:tyron:zil:test:EiAT_GxAt7gBozHlw2B1i7mfQaaORL3NOfFQr9FUt7jp6g?service=agentId1’, we can see that at the end we can be specific with a service we are trying to find such as agentID1 (Tyronzil, DID-URL Parameters). We can also use the version ID parameter in which searches for the DID document that contains a specific version similar to our example discussed earlier when it equalled ‘1’ to find the correct version number that resolves to ‘1’. This is similar to the third parameter that uses the version-time ID where we can find the specific date or timestamp that the DID document was resolved at. The format of this is usually specified by “month-day-year” or the UTC 00:00:00 standard time and can be easily seen in the URL in case of quickly changing it to be used faster and easier. Lastly, the last parameter is the ‘hl’ tag which is simply added to the DID URL for integrity protection. It is always specified as an ASCII string value so it can be incorporated into many of the popular programming languages such as Java and scripting languages such as Javascript. Like others including Tyron, some parameters need to include prefixes such as 'tyron-dns' to indicate other methods are being used that are not described here. Note that each of these identifiers for the DID and the DID URL follow the standard URI in the RFC Standard and can be easily referenced to those wanting to experiment and use DIDs or for developers incorporating DIDs into their programs for specific pieces of hardware. Now that I have discussed the main importance of identifiers in a manageable low-level way, it should be clear that DIDs can work relative to one another and some may have higher authority than the other. However, it does not change the fact that each DID identifies one specific entity and relates to one specific document. The Data Model of a DID can explain this and how it can be serialized into multiple concrete representations. In retrospect, a DID document consists of a map of entries in which each represents a key-value pair. According to W3C [Reference 2], each DID document data model consists of two types of classes which are core properties and representation-specific entities which can be described in *Figure 5* below (W3C, Section 4.1). Here we can see that the top portion of the figure represents the identifiers, methods, relationships, etc. along with each property extension. All of the properties in this section are always strings and can be represented by lists, maps, sets, datetime (simple date),

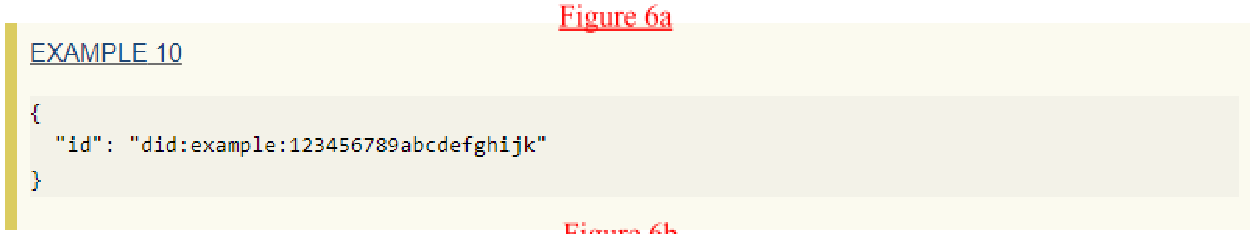
Figure 5


Entries in the DID Document map
In DID Specification Registries
Properties
Representation-specific entries
Core Properties
id, alsoKnownAs, controller, authentication, verificationMethod, service, serviceEndpoint, ...
Property Extensions
ethereumAddress
Unregistered Property Extensions
foo
Core Representation-specific Entries
@context
Representation-specific Entry Extensions
Unregistered Representation-specific Entry Extensions
%YAML, xmlns

strings, integers, doubles, booleans, and nulls. These are everything you will typically see in a normal programming language and here it is no different. The unregistered property extension is similar to that using other DID documents given from our path component mentioned earlier. The bottom of *Figure 4,* displays the representations and how each of the portions in the properties are incorporated into a physical form that is easier to read and interpret. This leads into the next section where I will explain the representations and the core properties of a DID and why it is very important that these properties are needed and or included in more detail. Although I have discussed identifiers, this section is important in that it will give more detail on how these can be improved for security considerations for future use and how developers can use these for including them in their code.

The DID core properties are all optional and all do not have to be included. Usually in a DID, the controller is the one specifying how to use or describe the subject or subjects it is referring to through the documentation. In the beginning of this segment discussing what a DID is in general terms I have described and given all of the relevant properties in *Figure 1* and talked about how each one of them are used. These include: id, controller, verificationMethod, authentication, assertionMethod, keyAgreement, and service. Each service and verification can be used in more detail and can be represented as lists or maps for in-depth use. The main idea is discussed through the DID Subject which takes advantage of the 'id' property and the DID Controller which takes control of the 'controller' property because these are the main ones users look for in a DID Document. From the Data Model in *Figure 5*, you can see that 'id' and 'controller' are the major core properties when categorizing and understanding a DID document. The DID Subject is always mentioned in the form shown in *Figure 6a* and the DID Controller is shown in the form given in *Figure 6b* given from W3C (W3C Section 5.1). The 'id' property must always be a string and exist in the root map of the data model given in *Figure 5* because it is necessary to explain the contents of the DID document. Usually, the 'id' is always found at the

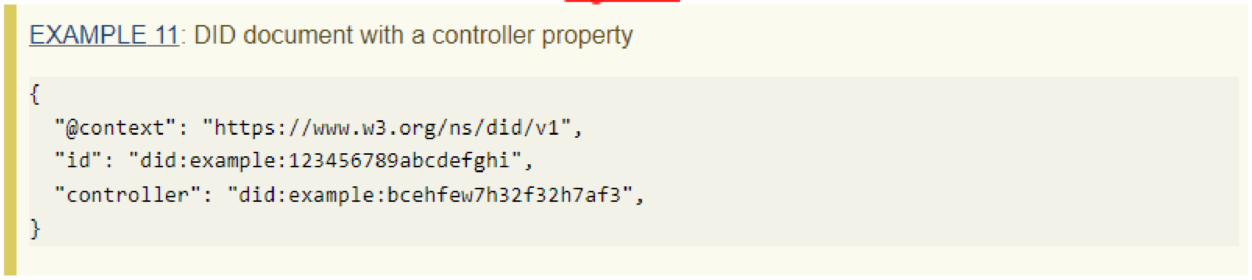
Figure 6a



EXAMPLE 10

```
{
  "id": "did:example:123456789abcdefghijk"
}
```

Figure 6b

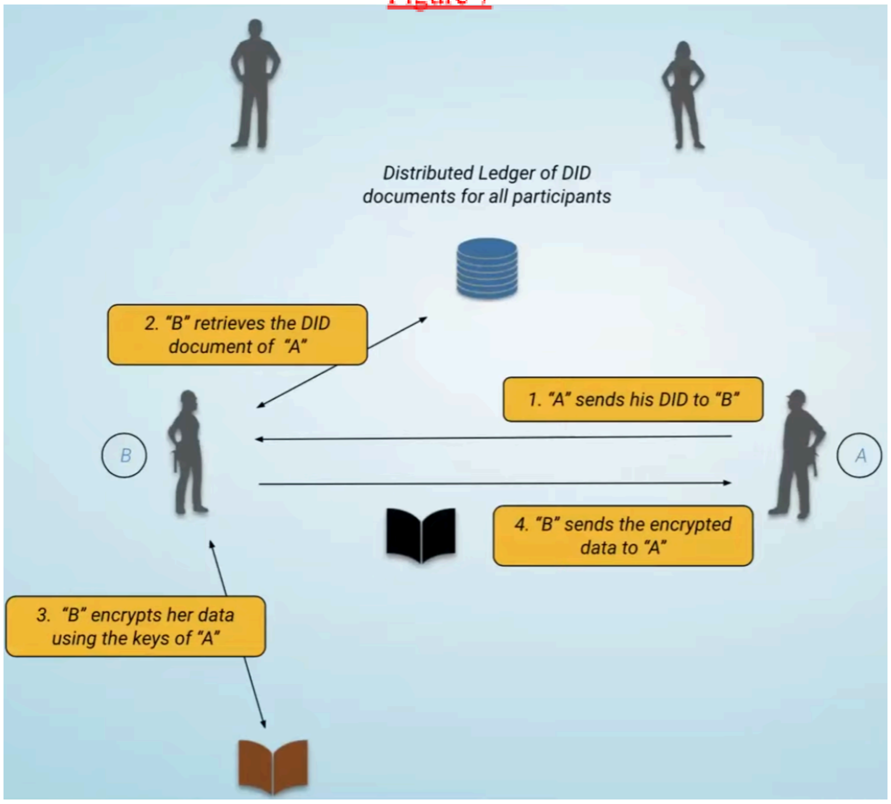


EXAMPLE 11: DID document with a controller property

```
{
  "@context": "https://www.w3.org/ns/did/v1",
  "id": "did:example:123456789abcdefghi",
  "controller": "did:example:bcehfew7h32f32h7af3",
}
```

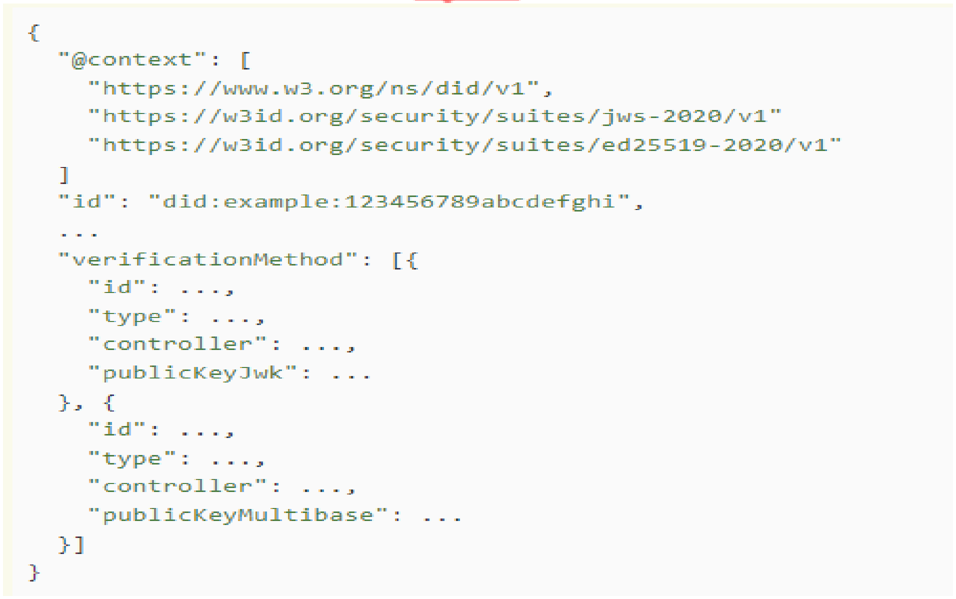
top of the document since it is super important and easy to initially grab. However, the 'controller' property is optional which means that a subject does not need a controller. A subject such as a bank asset for counting money does not have a specific controller similar to how a DID does not have to be controlled or changed to identify something. If a controller is present, similar to the 'id', it must be a string or a set of strings and it also should have verification methods or verification relationships (discussed after this) to ensure that the controller is indeed the correct user, machine, asset, etc. to edit or use the subject's property using the controller's public key. Recall that authentication is not the same thing as verification in which the controller only handles authentication, there is a separate property as you can see in *Figure 5's* core properties that have an authorization field for showing what entities can interact with the subject for specific purposes. The full picture is provided in *Figure 6b* where you can see the DID of the 'id' and 'controller' entity. The 'context' property is given so we understand a little more about the specific DID document and extra details for references. There is also an 'alsoKnownAs' optional property to identify multiple controllers if needed. However, let's say you are renting a car from a rental company and you need to provide a DID form of proof of a valid driver's license to the GRA (Government Registration Authority) to receive a document allowing you to rent from that company. In order to do so they need to have some way to verify your credentials through some type of verification method with public keys. From the W3C Fintech event held in 2020, Ivan Herman mentioned this as the 'Pool of Relationships' in his presentation [Reference 6] where he discusses that verification methods are included to ensure that encrypted details are being sent from one user initiating a DID exchange and the other verifying on a ledger or another source as discussed earlier (Herman, Slide 24). You can see from his presentation in *Figure 7* he provides a

Figure 7


Distributed Ledger of DID documents for all participants
2. "B" retrieves the DID document of "A"
1. "A" sends his DID to "B"
B
A
4. "B" sends the encrypted data to "A"
3. "B" encrypts her data using the keys of "A"

model that is easier to understand at a high level where he explains how the pool can be an easier way to capture the 'Ease of Use' goal through these type of verification methods instead of having both parties 'A' and 'B' known to each other. This ensures they can remain anonymous and even perform exchanges of DIDs through multiple ledgers (Herman, Slide 25). But why is it necessary to include a ‘verificationMethod’ attribute rather than having multiple controllers with another simple authentication attribute signifying who is and why they are accessing the document? Because a verification method is dynamic in which it can authorize interactions with the DID subject through a cryptographic public key with for example a digital signature so that signer can then use it as its cryptographic private key. According to W3C [Reference 2], we can see how the verificationMethod is usually written in *Figure 8* below. There are 3 sub-properties

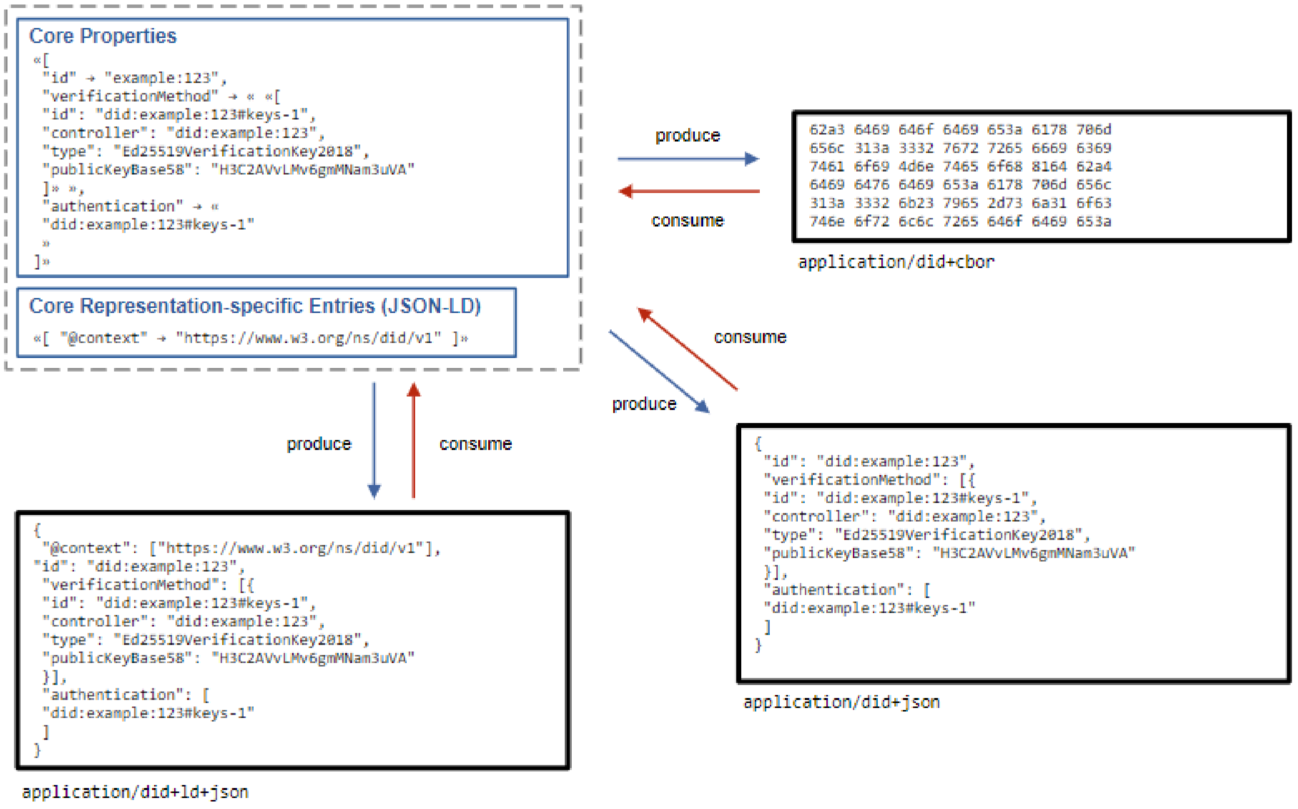
Figure 8



```
{
  "@context": [
    "https://www.w3.org/ns/did/v1",
    "https://w3id.org/security/suites/jws-2020/v1"
    "https://w3id.org/security/suites/ed25519-2020/v1"
  ]
  "id": "did:example:123456789abcdefghi",
  ...
  "verificationMethod": [{
    "id": ...,
    "type": ...,
    "controller": ...,
    "publicKeyJwk": ...
  }, {
    "id": ...,
    "type": ...,
    "controller": ...,
    "publicKeyMultibase": ...
  }]
}
```

to the ‘verificationMethod’ property that are ideally important. These are the ‘id’ which explains the holder of the public key, ‘type’ which tells us what verification type such as a proof, verification, or signature, the ‘controller’ who can modify the ‘id’’s holder of the public key such as a JSON Web Signature, and the public key itself that is underneath which contains the ‘id’’s public key. Notice that the public key can either be one of two common types: a ‘Jwk’ or a ‘Multibase’. A ‘publicKeyJwk’ maps to the subject’s JSON Web Key that has its own attributes and property names to safely secure it through its own cryptographic system. A ‘publicKeyMultibase’ is usually used for non-informative uses and doesn’t need extra information and security for verification. Usually these only contain a cryptographic public key and it simply just links to another peer in the ‘pool’ it is trying to connect with. These two keys are also optional, but it is really important to include these keys so we understand what individual is connected to the subject of the ‘verificationMethod’ and they are the correct user given from the public key. The next thing that is very important for discussion is the connection between the subject and the verification method itself. I mentioned earlier that the controller must be somewhere in the ‘verificationMethod’ but this only explains the controller, how do we know that the entities listed in the verification methods are in some form of fashion “related” to the subject? If we provide a DID to the rental company for my GRA document that gives me access to a rental vehicle, how will they know each verification subject can attest to my valid driver license and how can we establish a clear communication channel if something seems incorrect? This is discussed through verification relationships and the five curriculum to help establish this connection and provide more insight for each verification subject.. According to W3C [Reference 2], there are five extra properties that are not core properties that are included such as authentication, assertionMethod, keyAgreement, capabilityInvocation, and capabilityDelegation which are all optional and must be embedded in the verification methods (W3C, Section 5.3). First, authentication is a verifiable relationship simply used to specify how the DID subject is expected to be authenticated for instances such as logging in or performing challenge-response protocols. If the authentication goes through, then the DID method is allowed to execute and use that information. A DID method could add or delete keys, update the controller of the DID document, or even delete the document itself if it is not needed. Second, assertion is a verifiable relationship where it specifies what the DID subject is expected to express as claims such as issuing a VC (verifiable credential) similar to our rental car example. When the GRA provides us with a VC (document mentioned earlier) with our DID the rental car company checks if that VC has not been modified or revoked to be used to rent a vehicle. In other words, the ‘assertionMethod’ is used to assert proof that the GRA did in fact issue the VC so the DID subject can express the claim that they can in fact rent a vehicle. Third, key agreement is another verification relationship for peer-to-peer where it specifies how an entity can generate encryption material that contains confidential information for the DID subject. This makes sure that the verification method can establish a secure connection channel between the two parties to send this information. Lastly, capacityInvocation and capacityDelegation are both properties for verification relationships that allow the DID subject to either invoke cryptographic capability such as editing or altering other methods in the DID document, or using it for a specific purpose. This case is mainly helpful when discussing allowing access to use APIs on the web or the HTTP protocol. Delegation is helpful only for transferring usability to access the DID document for cryptographic capability such as a DID subject wanting to sign the capability to another DID subject through the capacityDelegation method. As you can see, these core properties are incredibly useful for a DID and overall explains what it is and how it is used in many different situations. It provides a more secure aspect than a simple identification system that is centralized because an attacker only needs to access the centralized database rather than more than one ledger for a DID. They will also have to understand how these five properties operate to protect an entity that is the subject of a DID. Verification works similar to that of service except verification methods as discussed above are used in sets and verify the DID subject whereas the service uses maps and allows the DID subject to communicate with other entities which are mainly used for advertising or other types of sharing of data. Now that the properties of a DID are familiar and how they operate and how a DID can be used in business and peer-to-peer transactions or secure channel communication, we need to quickly look at how a DID is commonly represented. According to W3C [Reference 2], the two representations used by DIDs are through JSON or JSON-LD but can be expressed as others through production and consumption (W3C, Section 6.1). Production occurs when we serialize the data model given to us from *Figure 5* into a JSON or JSON-LD and consumption works the opposite where we want to turn a representation to a data model so maybe we can translate between representations such as YAML or XML. This makes it so much easier for developers to work with and produce a common data model that each developer can understand. This idea is very familiar to us through UML (Unified Modeling Language) diagrams where we can convert UML 'Use Cases' for instance into pseudo code and then executable code except in this case, we are going from a common data model to JSON, JSON-LD, or anything that is similar for a developer that doesn't know how to process specific representations. *Figure 9* below represents this idea where W3C [Reference 2] explains the idea of producing JSON files from the data model along with other common representations for others to use for each DID (W3C, Section 6.1) . As you can see we

Figure 9



```
«[
 "id" → "example:123",
 "verificationMethod" → « «[
 "id": "did:example:123#keys-1",
 "controller": "did:example:123",
 "type": "Ed25519VerificationKey2018",
 "publicKeyBase58": "H3C2AVvLMv6gmMNam3uVA"
 ]» »,
 "authentication" → «
 "did:example:123#keys-1"
 »
]»
```

```
«[ "@context" → "https://www.w3.org/ns/did/v1" ]»
```

```
62a3 6469 646f 6469 653a 6178 706d
656c 313a 3332 7672 7265 6669 6369
7461 6f69 4d6e 7465 6f68 8164 62a4
6469 6476 6469 653a 6178 706d 656c
313a 3332 6b23 7965 2d73 6a31 6f63
746e 6f72 6c6c 7265 646f 6469 653a
```

application/did+cbor

```
{
 "id": "did:example:123",
 "verificationMethod": [{
 "id": "did:example:123#keys-1",
 "controller": "did:example:123",
 "type": "Ed25519VerificationKey2018",
 "publicKeyBase58": "H3C2AVvLMv6gmMNam3uVA"
 }],
 "authentication": [
 "did:example:123#keys-1"
 ]
}
```

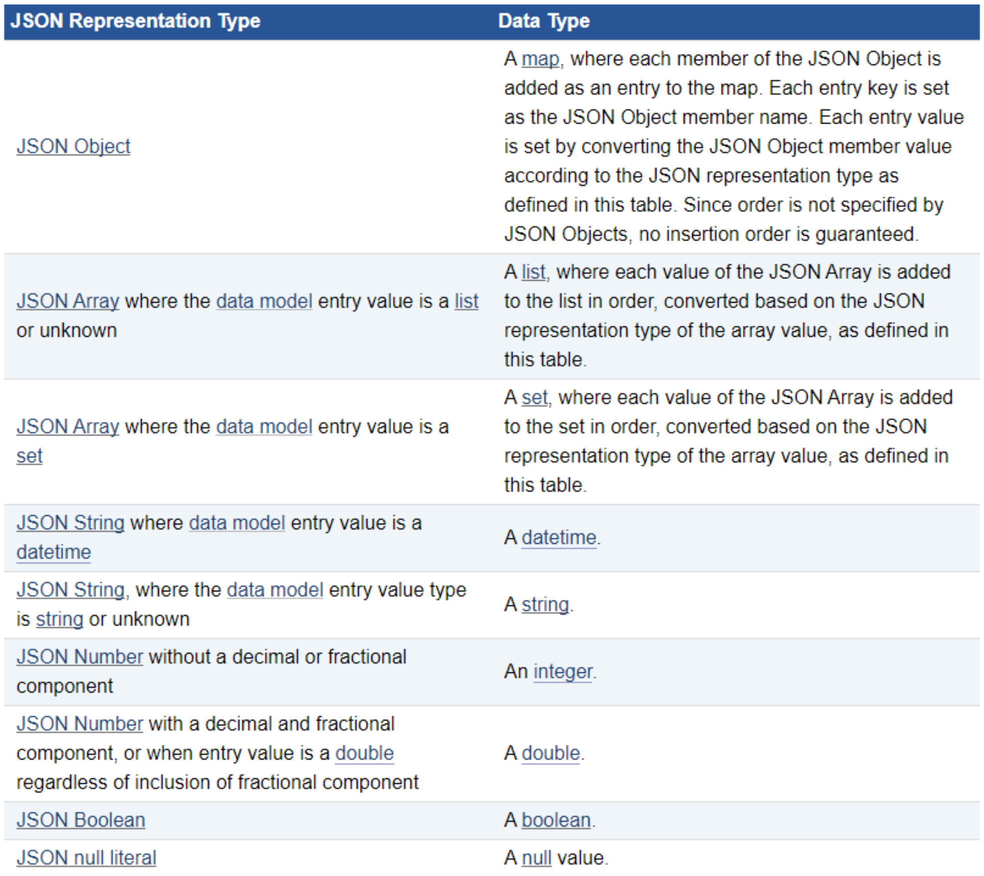
application/did+json

```
{
 "@context": ["https://www.w3.org/ns/did/v1"],
"id": "did:example:123",
 "verificationMethod": [{
 "id": "did:example:123#keys-1",
 "controller": "did:example:123",
 "type": "Ed25519VerificationKey2018",
 "publicKeyBase58": "H3C2AVvLMv6gmMNam3uVA"
 }],
 "authentication": [
 "did:example:123#keys-1"
 ]
}
```

application/did+ld+json

convert CBOR (binary) into JSON-LD (linking-data) by first consuming into a common data model and then translating it to JSON-LD by production. There are many rules that go along with production and consumption of different representations. For a JSON representation has their own specific representation types for production rules and consumption rules. This is shown in *Figure 10* below that is given from W3C [Reference 2] (W3C Section 6.2). Notice that here we can also infer consumption. A map goes to a JSON Object, and a JSON Object converts back into a map in the data model. So you can see translating the information in a DID document is not that hard when each representation follows the idea of production and consumption. It is

Figure 10



JSON Representation Type	Data Type
JSON Object	A map, where each member of the JSON Object is added as an entry to the map. Each entry key is set as the JSON Object member name. Each entry value is set by converting the JSON Object member value according to the JSON representation type as defined in this table. Since order is not specified by JSON Objects, no insertion order is guaranteed.
JSON Array where the data model entry value is a list or unknown	A list, where each value of the JSON Array is added to the list in order, converted based on the JSON representation type of the array value, as defined in this table.
JSON Array where the data model entry value is a set	A set, where each value of the JSON Array is added to the set in order, converted based on the JSON representation type of the array value, as defined in this table.
JSON String where data model entry value is a datetime	A datetime.
JSON String, where the data model entry value type is string or unknown	A string.
JSON Number without a decimal or fractional component	An integer.
JSON Number with a decimal and fractional component, or when entry value is a double regardless of inclusion of fractional component	A double.
JSON Boolean	A boolean.
JSON null literal	A null value.

good to note that JSON-LD also follows this pattern, however, it has an extra '@context' property (mentioned earlier) that gives a little more context what the entity is supposed to do through combinations of strings and other integers or even null values! Although some representations require a little more effort, they are all very common because each DID supports a similar data model where each conversion can happen such as a JSON Number corresponding to a data model double which corresponds to a YAML Number. Through this, we have discovered how a DID is modeled through its documentation and how they are represented. As we complete the puzzle of how a DID works, what it is, and how it can replace the many identifying struggling factors in our daily lives, it is also necessary to look at the privacy benefits of DID and how it can change our identification system to prove a lot more efficient. The next segment, I will discuss the core benefits of DIDs and some downsides of what the DID can possibly do.

3 - Benefits of Decentralized Identifiers

In this segment, I will be discussing the major three benefits of a DID and one downside that can possibly cause concern for the future. Although a decentralized identification system would make things a lot easier, it doesn't exclude the fact that it can still cause problems for people down the line. In some instances, it is better to weigh the odds of what a DID can provide in terms of utility and see if the drawbacks outweigh its utility and then discuss possible reasons why to assist in solving this problem in the future. When discussing the benefits some will view one side as a chance to make identification easier to use while others might think it will increase fraud or ruin the true identities of individuals. This can be related back to the discussion of how Bitcoin and other cryptocurrencies came about. Many saw Bitcoin as destructive and the evidential fall to the world's economy because it was decentralized and not ordered through our government. Some others saw huge problems with ledgers and the cryptography involved in hashing specific important information in Bitcoin so other cryptocurrencies came about such as

Litecoin, and Ethereum. Although an individual has a choice to invest in what cryptocurrency that interests them, it should be similar to that of decentralized identification systems. Through this section, I will discuss each benefit of why DIDs should be continued to be implemented but also a concern to take into account when developing DIDs.

The first major benefit of DIDs is that they are very secure and private that prevents fraudulent behavior significantly compared to centralized identification systems. In the past decade, we have seen the development of MFA (multi-factor authentication) being developed in various ways to prevent bad actors from infiltrating systems or accounts registered in the system. The main reason behind this is due to the untrustworthiness in security in most current generation technology. This idea doesn't always have to be systems or software themselves either, in fact, there are a lot more fraudulent attempts that stem from other verification systems such as forging signatures, or pretending to hold a certificate or a degree when the bad actor does not actually possess it. The benefit of security is protecting small business owners that are either startups or even large companies with staff that can use DIDs to prove one's credentials actually exist. For instance, as an undergraduate, applying for a full time job requires me to verify that I have completed my degree along with a lot more documents that contain who I am, all security checks which can all be resolved into DIDs. According to Amarachi Johnson-Ubah [Reference 7], a web-developer discusses the need of how ex-Twitter CEO Jack Dorsey discusses the need to push toward something called "web 5" rather than "web 3" in which it empowers users to control their own centralized identity to meet the common goals discussed in the introduction to this report such as 'Ease of Use' and being 'Persistent' (Ubah, Benefit Section). Abuh also goes on to discuss that decreasing fraud using DIDs can provide us with quicker and convenient Verifiable Credentials for individuals to reduce cost and efficiency while still maintaining privacy for the user or entity it represents. If a user wants to have a digital identity for a specific organization like Amazon, they can do so by generating their own cryptographic key that is stored on the blockchain (or ledger, depending on methods discussed in the previous segment), which is immutable because it is stored on many computers around the world. This also influences privacy which gives the user a right to permit and gives them the ability to choose what information they want to disclose and or provide an application for access. One common inquiry is the process and how the blockchain prevents this criminal activity. More specifically, although we store our identities and recorded information on the blockchain, what will happen if something is altered or changed, or if some fraudulent activity actually happens? Recall in the previous section, we discussed how each property of a DID is usually encrypted with some type of method provided to us by the 'type' property along with other 'verificationMethod' properties for the controller or those who want more security in DID documentation. If these are altered, it can cause damaging effects of the holder of the DID. According to W3C [Reference 2], they provide a complete list of recommendations and ways to secure a DID from alteration such as authentication service endpoints, proving control and binding of DIDs, and even making some DID methods completely immutable (W3C, Section 9). However, one thing that stands out is the verification method rotation technique along with the key and signature expiration. Verification method rotation is the process that enables "the secret cryptographic material associated with an existing verification method to be deactivated or destroyed once a new verification method has been added to the DID document" (W3C, Section 9.7). What this means is that when a new verification method is produced, it has to reiterate the process of verification to make sure the whole DID document, including the controller, has a valid DID, DID Method, and contains a valid identifier. This is useful because it can protect against verification compromises since it reduces the value of a single verification method to be released to an attacker. Of course, it is a good idea to rotate the verification methods on a daily basis, along with ensuring that the holder of the DID (DID subject) is not frustrated in having to refresh their associated credentials. This type of verification rotation is handy for security since it gives holders a way to protect their data in a more secure way than most current forms of MFA and without having to go through the process of verifying with the data registry (ledger). The next security protocol that can be implemented to a DID that stood out to me is the key and signature expiration. What this does is it uses DID resolvers which are separate pieces of software that perform security checks before resolving a DID method into the actual DID document when it is initially created (W3C, Section 9.6). Therefore, these resolves can use verification libraries and separate software to validate materials in the DID documentation along with users requesting their own expiration policies. This can also work with incorporating and extending the use of digital signatures to be also used as a key that can expire. As you can infer, these safety methods can increase the result of safe identification techniques between 2 parties whether it be for business or even just peer-to-peer. There are plenty more security measures that are planned to be executed in the future. One of these, according to W3C [Reference 2] is Notification of DID Document changes such as the subject changing to prevent account takeovers (W3C, Section 9.5). There are many other security measures that also protect privacy that can be implemented so there shouldn't be a huge worry of security measures. This doesn't mean, however, that security measures should be standard to a specific DID and rushed to complete a security protocol that is very vulnerable to attackers. The benefit of these existing DID security methods is to ensure that it meets its goal of providing 'cryptographically verifiable' and 'resolvable' measures so attackers don't cause havoc. It is a very good benefit since security is something that should be taken seriously and developed in mind without being developed on the side. Security ultimately increases the privacy of users who are using the DIDs for their own subjects or for their own benefit for even very private matters and it would be extremely detrimental if this information leaked or infiltrated by attackers.

The next benefit of DIDs goes back to our original goals discussed previously on the 'Ease of Use'. In fact, the biggest general benefit of DIDs is that they make data extremely portable without the need to remember certain passwords, security related questions, authenticity MFA, or even biometrics if needed since it is stored wholefully within DID documentation for that subject. According to Amarachi Johnson-Ubah [Reference 6], most identifiers and attestations are locked into the database of the issuing organization which means that you have to constantly use your set of credentials for that application only that specific time and you can not use the same for some other application (Ubah, Benefits Section). The importance of DID lies in the fact that it is extremely easy to use due to the portability. Let's take an example and say that we want to sign up for a grocery store application in order to make checkouts easier and a lot faster without having to scan every single item. Normally, a user would simply open the application on their phone, create a username and password, and then have to create an MFA to make sure their account details are safe, and then they would have to go through verification steps like verifying their email. Once this is complete, they then need to add some type of payment method into the application in order to pay and check out a lot faster and they also might need to add their grocery list and include their car model and make it if they want to do pickup. As you can see, this is a lot to do especially if a user wants 'Ease of Use' and a more 'Decentralized' system. Using DIDs is simple enough to combat this. Ubah [Reference 7] mentions that having to create an account per application on the web or on the phone is completely unnecessary because then you will also have to remember your login credentials and authentication methods. With using a DID for credentials in a controllers digital wallet, they will be able to manage their own authorizations, IDs and data for varying applications, sign-in credentials for applications or services that do not have a decentralized system, as well as adding car information details such as the year, model, make and the payment information. All this information is simply stored on the users DIDs for either payment information and common login information so they can immediately sign up to the grocery store application and begin using it without having to go through the initial steps of creating an account and verifying that they are indeed the user of the application. As mentioned previously, W3C [Reference 2] mentioned that parts of the DID documentation can be editable by the controller given the standard verification methods and therefore can be versatile and adaptable to change given that each verification method is correct (W3C, Section 5.2). This means that if the user needs to change information for something such as their plate numbers on their car, they will be able to do so if they have a specific DID for the car registration on another DID for their credentials to sign up. Many DID documentations can refer to other DID documents where they are absolutely necessary so it categorizes the information and it is 'Resolvable' rather than having to reiterate through one DID and then the other. This also works in the tangent of adding multiple accounts to a DID because it is stored in your digital wallet. You essentially do not have to sign up for everything because the system already understands you and can see how you directly interact with services for vendors or businesses. This is what makes DIDs portable, in which you do not have to rely on a third party to conduct an investigation or provide a service for you but instead you are able to directly access the investigation or service. Doing it like this is what makes decentralization a smart solution to the troubles of storing passwords and remembering all your credentials when you can simply carry it with you. If past credentials are not used anymore such as a service that goes out of business, you will still be able to maintain all the data you have stored due to your decentralized identity. This can also be meant in protecting you from bad actors in that data is portable and constantly updating on a data registry, so security assets are always attached and extremely viable. The benefit of having your identity easily verified, proven, and portable is always considered since it provides a much more viable option then the simple identification methods that we have today.

The last benefit of DIDs that I think is very useful is the advantages for the organizations as I have already analyzed most of the benefits within an individual. One of the biggest benefits of DIDs for these organizations is that it is a very speedable process at a very low cost due to not having extra software to manage a centralized database. According to Dock [Reference 8], an actual company that implements commercial DID software (more on this later), stated that more data breaches to these central database servers are very costly to almost $108,000 in damages for small businesses and takes about 73 days to fix them (Dock, Key Benefits Section). Of course, it is very important to include Cybersecurity tactics in a small business, however along with having a dedicated SOC (Security Operations Center) and information security engineer team, it can be very beneficial to decentralize certain assets in order to make them DIDs to increase the security not only for users of that small business but also the assets themselves. Increasing the security is very important but it also makes it super beneficial that it makes it easier on software developers as well. According to Dock [Reference 8], creating user-centric apps that eliminates the need for passwords as well as authentication processes will not only enhance the user experience, but it will also make it very easy for a developer to maintain (Dock, Key Benefits Section). Overall, this reduces the costs of the management of a centralized database doing its job properly and not subject to cyber threats. Another big benefit for small businesses and organizations relative to cost is that users are able to request data directly without having to worry about privacy exceptions within an application. For instance, if a user wants to request a loan from the bank they do not need to hold all the relevant paperwork that requires documentation about their private information such as their salary, address, etc. rather developers working on the bank can use their DID and verify those private credentials directly instead of having to request data to verify them. In general, an organization can easily encrypt private information and decrypt that information safely without having to worry about outdated verification and privacy details being held centrally rather than a ledger. These are some of the most important benefits that explain what a DID can do to make daily tasks for us easier while still incorporating the necessary tools such as security and privacy, ease of use for portability, and of course advantages for businesses and developers. However, there are disadvantages of using DIDs that can make it hard to fix without risking user data and creating concerns with how that data is being held.

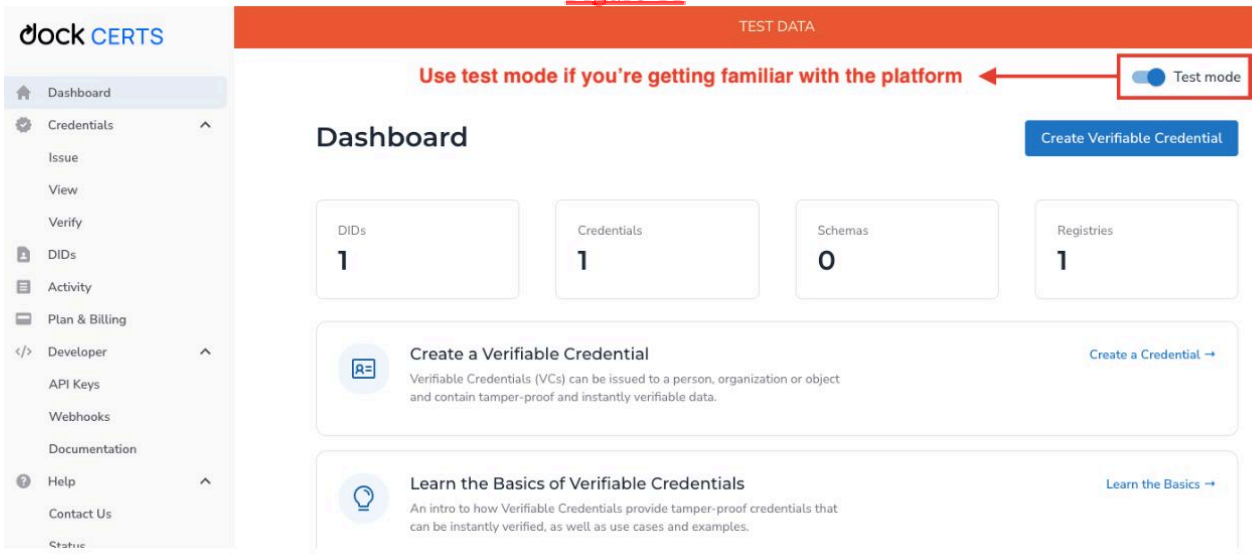
One concern for DIDs is simply the "digital wallet" or where and what information you are allowing third party software to hold. In other words, the problem is that in order to take advantage of DIDs we still have to use some type of software to access them and it can be dangerous because that third party can still be threatened by attacks of the commercial software themselves or the consent of data being revoked. According to the Identity Management Institute (IMI) [Reference 9], they described a person may grant access for a DID while the consent of the data may be revoked in the future or identity-self management can be threatened on the blockchain (IMI, DID Management Risks Section). If someone decides to purchase a good and later revokes it, it can cause serious issues across multiple ledgers since these bad actors are already verified through the verification methods in the DID documentation. This means that these bad actors can still commit fraud although it's through the data registry. All in all, even though DIDs need to have security measures as mentioned previously in this report, it is still necessary that third-party software or digital wallets implement their own security methods to reduce these kinds of risks. Still, these can be expensive and tedious which also switches the target from the actual DIDs and companies to the actual software containing the DIDs. Of course, when DIDs become more and more developed overtime, we will still eventually find ways to close the issue of threats to digital wallets or consent issues. There are even some companies such as Dock who release commercial DID software to be used daily which leads into the next discussion in the next segment about some of the features of these DID software products and how they are used with some examples.

4 - Commercial and Research Software

As of now, there are many DID software products that are being used daily in order for individuals, businesses, and developers to verify their identities whether it be through themselves, the products they are selling, or any other entity related to their business. In this segment, I will be talking about the features of what you can do and how you use these DID software products including the strong and weak features of each. The main commercial software products I will be exploring are Dock Certs, Microsoft Entra, and Trinsic Dentity.

Dock Certs [Reference 8] is a type of DID commercial and research software created by Dock which focuses on developing software to create a safe digital identity. They work in many industries such as healthcare, supply chain, finance, and education. Dock's main goal in their software is to create a secure verifiable credential that can be used for proving that the user owns or has something such as an account or a valid document in order to make verification a lot easier. Although these DIDs aren't mainly for small businesses, they are still heavily tied to the individual or developer because they provide many different ways to issue, view, or verify credentials of a user for various purposes. One of the strong features of this software is that you can actually test out different aspects of how a DID works such as creating an issuer profile or view and create verifiable credentials (testing credentials). *Figure 11* below portrays a screenshot of the Dock Certs dashboard where you can fully interact with your credentials or test to see how to add 'dummy' credentials using its dedicated test mode (Dock, Creation Section). Also, notice

Figure 11



Dashboard

DIDs	Credentials	Schemas	Registries
1	1	0	1

Create a Verifiable Credential

Verifiable Credentials (VCs) can be issued to a person, organization or object and contain tamper-proof and instantly verifiable data.

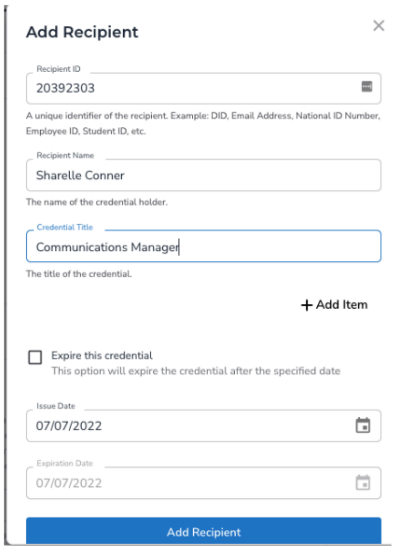
Learn the Basics of Verifiable Credentials

An intro to how Verifiable Credentials provide tamper-proof credentials that can be instantly verified, as well as use cases and examples.

that Dock Certs provides a lot of ways you can learn about the basics of verifiable credentials in which it will take the user through a step by step process of how to create a DID. Dock Certs offers different templates such as a basic credential or a CV to be added along with adding recipients using their ID they used when they signed up with Dock. You also have options to expire the credential (we discussed this previously) and provide the issue date to keep track later if something happens or in case the holder partakes in some type of fraudulent behavior. This is another strong feature because just in case in the future that you don't want another user to have access to your credential, you can always make it so it gets rid of it from being used elsewhere.

Figure 12 below shows how you would add a recipient to your DID, notice to do so, you need the recipients 'id attribute in which each DID subject has that is using another DID for their verification. One other really useful feature that Dock Certs has is that you can add multiple

Figure 12


Add Recipient

Recipient ID
20392303
A unique identifier of the recipient. Example: DID, Email Address, National ID Number, Employee ID, Student ID, etc.

Recipient Name
Sharelle Conner
The name of the credential holder.

Credential Title
Communications Manager
The title of the credential.

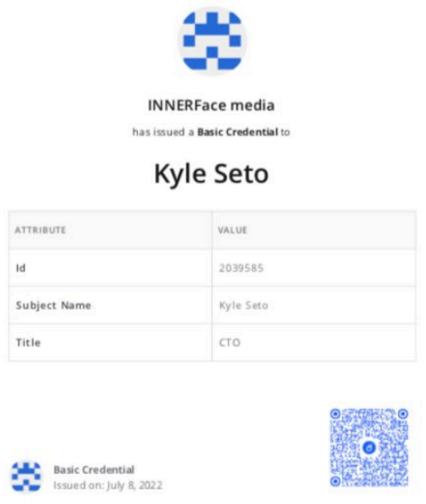
Expire this credential
This option will expire the credential after the specified date

Issue Date
07/07/2022

Expiration Date
07/07/2022

credentials for a recipient for further use. This means that you can share multiple credentials (DIDs) if you have them guaranteed by the issuer. There are two more really strong features that Dock provides the user. The first is that you can 'persist' the credential which means that instead of storing your DID on the blockchain, it will go to Dock's own database server that is also highly encrypted. Don't be confused about Dock's persistent feature with the 'Persistent' goal of a DID since they are not similar at all. If a user decides not to use Dock's persistent feature, then the credentials will be stored on the blockchain and it will also be stored on Dock's database. This leads to the second strong feature that if the user is also using Dock Certs and not any other DID software, then you can easily scan the credential through the app with the barcode that is provided on the credential. If you look at *Figure 13* below, you will see that at the bottom of the created credential there is a barcode to be scanned through the Dock Certs app which it will then check to see if it is verifiable. Keep in mind that Dock Certs is still experimenting with different features and taking advantage of everything that DIDs have to offer. However, there are some

Figure 13



INNERFace media

has issued a **Basic Credential** to

Kyle Seto

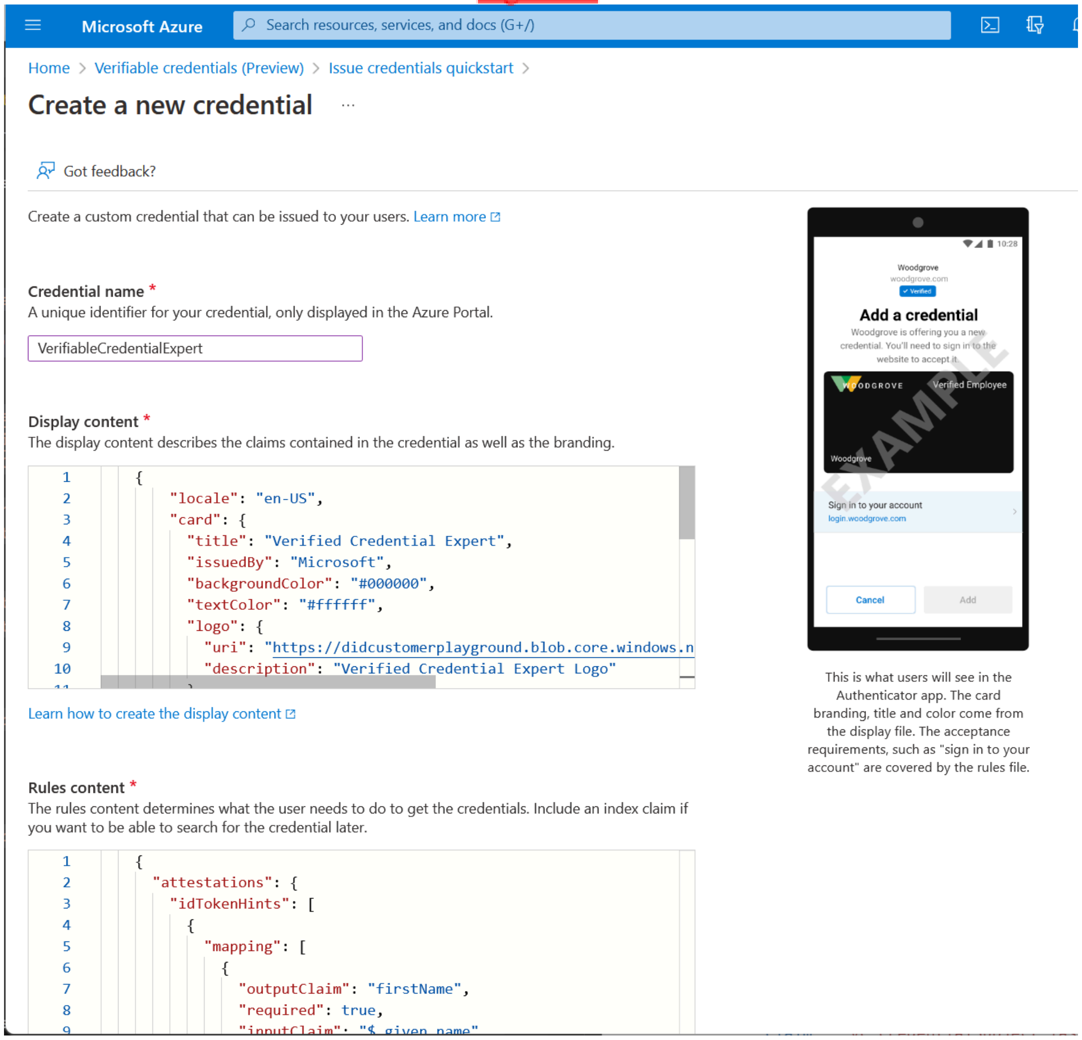
ATTRIBUTE	VALUE
Id	2039585
Subject Name	Kyle Seto
Title	CTO

Basic Credential
Issued on: July 8, 2022

weaknesses such as requesting for credential tiles and templates along with key IDs. For instance, to use Docks service as an issuer, say at a University issuing a college degree to a student who recently graduated, they have to first verify through Dock that they can issue the credential. It would be a pain to constantly check for templates from Dock themselves which almost diminishes the 'Decentralized' aspect of it. To make it more proper, they can create some sort of AI that verifies that a University is issuing a DID to a student. The next weakness to these features is that since it is also storing your data (encrypted of course) on Dock's database as well, it costs money for plans. Although they are not super expensive, it will still be a headache for a small business or a university to sign up for these plans. Next, I will discuss Microsoft's DID software which is called Microsoft Entra and is made to be more secure.

Microsoft Entra [Reference 10] is a DID software that is unique in which it also issues credentials but contains the same security and privacy features of that offered through Microsoft Azure. Similarly to Dock, Microsoft Entra also stores records and user data that is encrypted on the Azure Active Directory. This means that even though data is passed through on the blockchain, it is also stored in the directory. Another benefit of Microsoft Entra is that the user can select anyway of how they want to issue the credential. This means they can import code or type up code to exactly parameterize each property in a DID similarly how I talked about it previously. The user can customize everything from rules, to expiration and even incorporate other digital wallets to use their DIDs through Microsoft Entra. *Figure 14* displays a screenshot of the actual software when creating the DID, notice that you can provide a multitude of different ways to fully customize the credential with even the card it will be displayed on (Microsoft Entra, Issuing Section). Another interesting benefit is that you may also customize the way to

Figure 14


Microsoft Azure
Search resources, services, and docs (G+/)
Home > Verifiable credentials (Preview) > Issue credentials quickstart >

Create a new credential ...

Create a custom credential that can be issued to your users. Learn more

Credential name *
A unique identifier for your credential, only displayed in the Azure Portal.

VerifiableCredentialExpert

Display content *
The display content describes the claims contained in the credential as well as the branding.

```
{
  "locale": "en-US",
  "card": {
    "title": "Verified Credential Expert",
    "issuedBy": "Microsoft",
    "backgroundColor": "#000000",
    "textColor": "#ffffff",
    "logo": {
      "uri": "https://didcustomerplayground.blob.core.windows.n
      "description": "Verified Credential Expert Logo"
```

Learn how to create the display content

Rules content *
The rules content determines what the user needs to do to get the credentials. Include an index claim if you want to be able to search for the credential later.

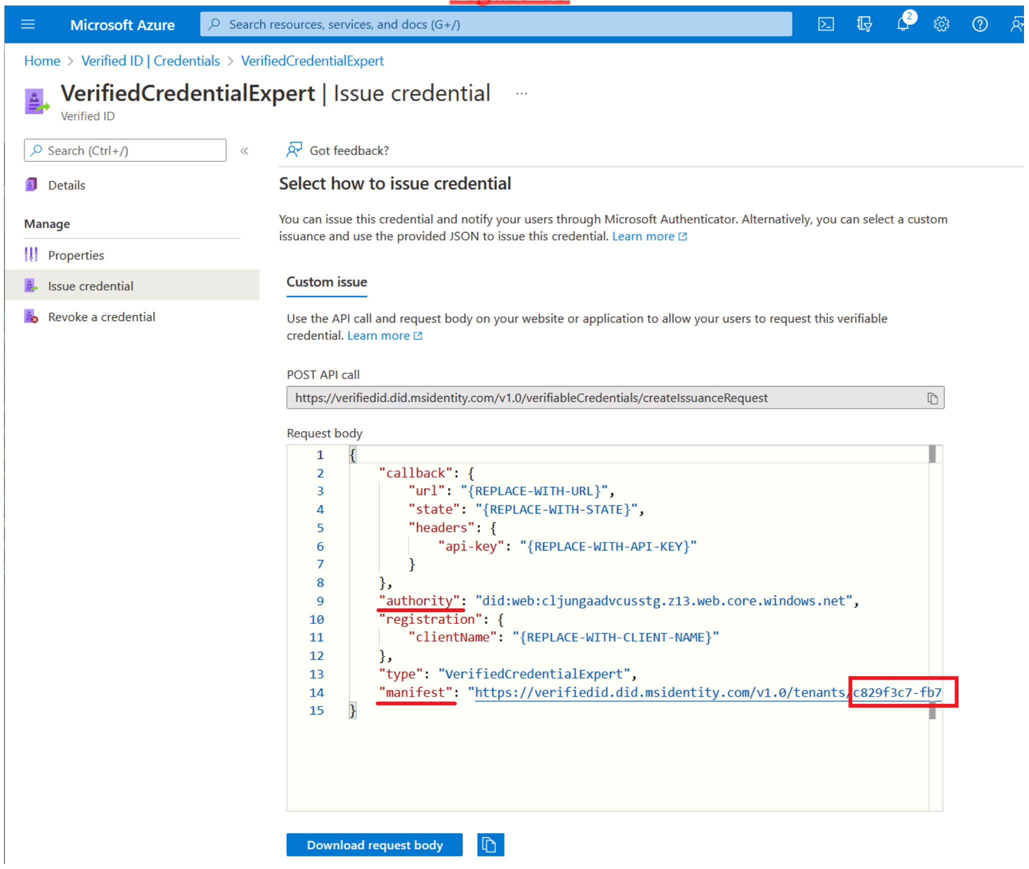
```
{
  "attestations": {
    "idTokenHints": [
      {
        "mapping": [
          {
            "outputClaim": "firstName",
            "required": true,
            "inputClaim": "$.given_name"
```

This is what users will see in the Authenticator app. The card branding, title and color come from the display file. The acceptance requirements, such as "sign in to your account" are covered by the rules file.

issue your credentials. Unlike Dock Certs, the user can provide a custom issue and use certain

APIs to post the credentials. Even though Microsoft uses different syntax for their JSON files for each specific DID, it is relatively similar in which they both like I mentioned earlier, in the data model can converge to be translated. The way a user can add properties to their issued DIDs is shown in *Figure 15* below in the case the user wants to issue in a custom environment with their

Figure 15


Microsoft Azure
Search resources, services, and docs (G+/)
Home > Verified ID | Credentials > VerifiedCredentialExpert

VerifiedCredentialExpert | Issue credential

Verified ID

Search (Ctrl+/)
Details
Manage
Properties
Issue credential
Revoke a credential
Got feedback?

Select how to issue credential

You can issue this credential and notify your users through Microsoft Authenticator. Alternatively, you can select a custom issuance and use the provided JSON to issue this credential. Learn more

Custom issue

Use the API call and request body on your website or application to allow your users to request this verifiable credential. Learn more

POST API call

https://verifiedid.did.msidentity.com/v1.0/verifiableCredentials/createIssuanceRequest

Request body

```
{
    "callback": {
        "url": "{REPLACE-WITH-URL}",
        "state": "{REPLACE-WITH-STATE}",
        "headers": {
            "api-key": "{REPLACE-WITH-API-KEY}"
        }
    },
    "authority": "did:web:cljungaadvcusstg.z13.web.core.windows.net",
    "registration": {
        "clientName": "{REPLACE-WITH-CLIENT-NAME}"
    },
    "type": "VerifiedCredentialExpert",
    "manifest": "https://verifiedid.did.msidentity.com/v1.0/tenants/c829f3c7-fb7
}
```

own API. This way, any user can send the credential to anywhere and any digital wallet they want to. Another really beneficial feature that Microsoft Entra offers is connecting Microsoft Authenticator which is an application on your mobile device to create a DID card similar to Dock to use for authentication of your identity. The way this is done is very confusing and requires 3rd party software such as ‘ngrok' in order to be able to get an expert card through Microsoft Authenticator. This is considered one of the downsides of Microsoft Entra. Although it provides a lot of customizability for accessing and using DIDs, it doesn’t help the fact that it can’t be easy to use since it relies on bringing the application to ‘get the credential’ on the local host machine. *Figure 16a* and *Figure 16b* displays when the user goes through a tedious process of issuing the credential through Microsoft Studio Code and then linking all the files from Azure to ‘ngrok’ to bring it to the local host’s or the user’s machine. Once they have added it they will be able to scan the barcode using Microsoft Authenticator and type the pin code into their mobile phone to confirm that they can be issued the credential. Once they get the credential on their phone they use it to provide identification to services given that it is verified and it does not expire. Choosing which DID software to use can be complicated, but each has their own benefits

Figure 16a

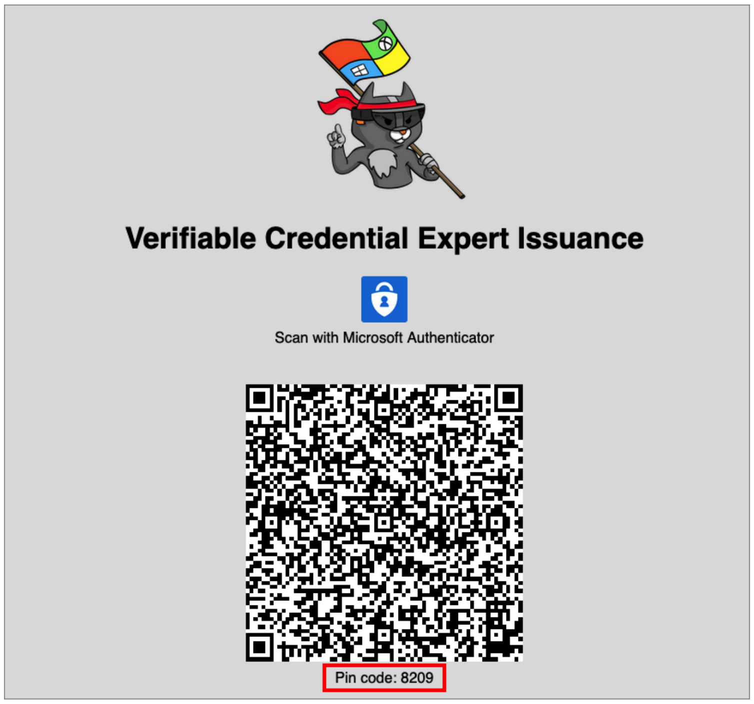
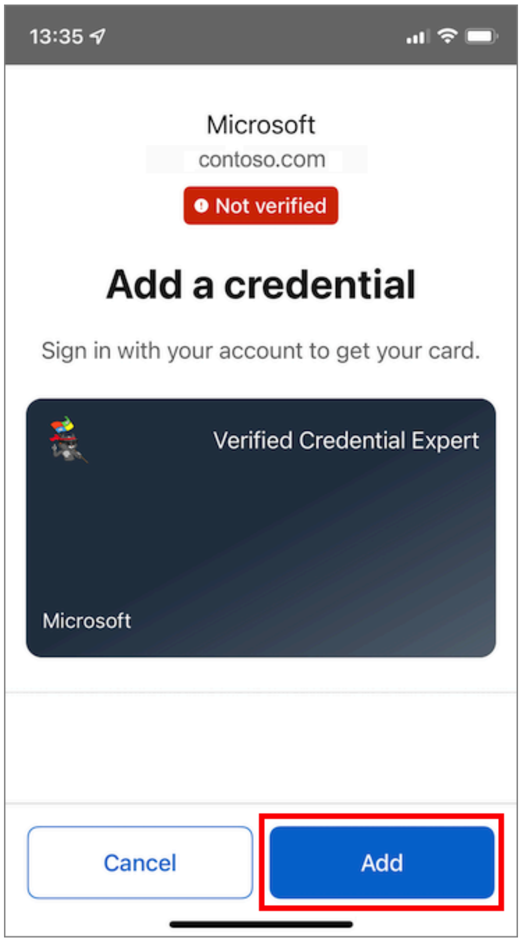

Verifiable Credential Expert Issuance
Scan with Microsoft Authenticator
Pin code: 8209

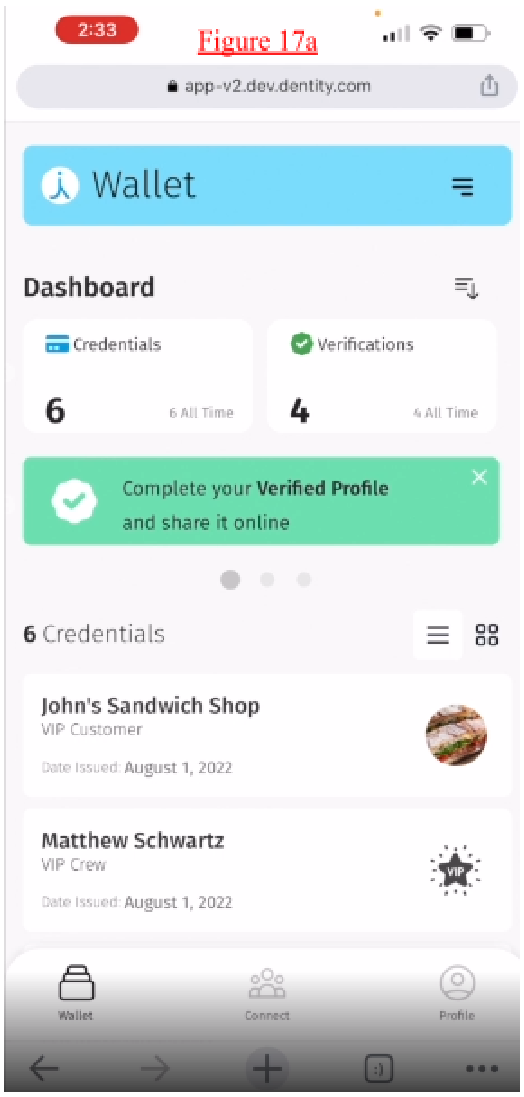
Figure 16b



and weaknesses regarding the simplicity or security. Microsoft Entra allows a safe and completely credible service while Dock has a more restrictive service but it is less versatile in which you can’t completely customize everything you need on a DID document. Since DIDs are expanding, DID software will be able to push past developer use and will let individuals who don't understand how they operate to create DIDs. Next I will discuss one more commercial product Trinsic Dentity that allows easier use for individuals that need to provide credentials in a much more simplistic way.

The last and simple DID commercial software that I think is worth mentioning is Trinsic Dentity [Reference 11]. This software works similar to that of Dock Certs but it makes actions a little more simplistic in which the user can directly scan a credential from any other digital wallet to be stored on your own (Trinsic, Web3 DID Section). For instance, if an issuer on another DID software issues a credential for a specific VIP service, then they will be able to add that credential with that specific barcode to the blockchain and the recipient's wallet. Another really big feature of Trinsic Dentity is that you can create an Open-ID Connection Application which is essentially having multiple connections to the issuer, similar to that of a multi-issuer where if an organization partners with another company, you will be able to access not only the company that you have a verifiable credential for, but also for the partnering company information that your credential is valid for (Trinsic, DID Presentation). In order for a company to do this, they will need the client ID from the user which can be a DID within itself, the client secret ID for authentication, and the redirect links and logout links for the specific user with valid credentials. *Figure 17a* portrays the dashboard of Trinsic Dentity from a simple demonstration presentation. Notice that you can create a public profile, to make transactions of credentials easier which is interesting since it increases peer-to-peer relations rather than just a issuer and holder for a small business and a peer who needs these credentials. *Figure 17b* shows the public profile in which you can create to be displayed publicly if needed. This reduces the time for a specific user who needs proof of identification to request access to the specific DID. However, one thing that is not

Figure 17a



2:33

app-v2.dev.dentity.com

Wallet

Dashboard

Credentials 6 6 All Time

Verifications 4 4 All Time

Complete your **Verified Profile** and share it online

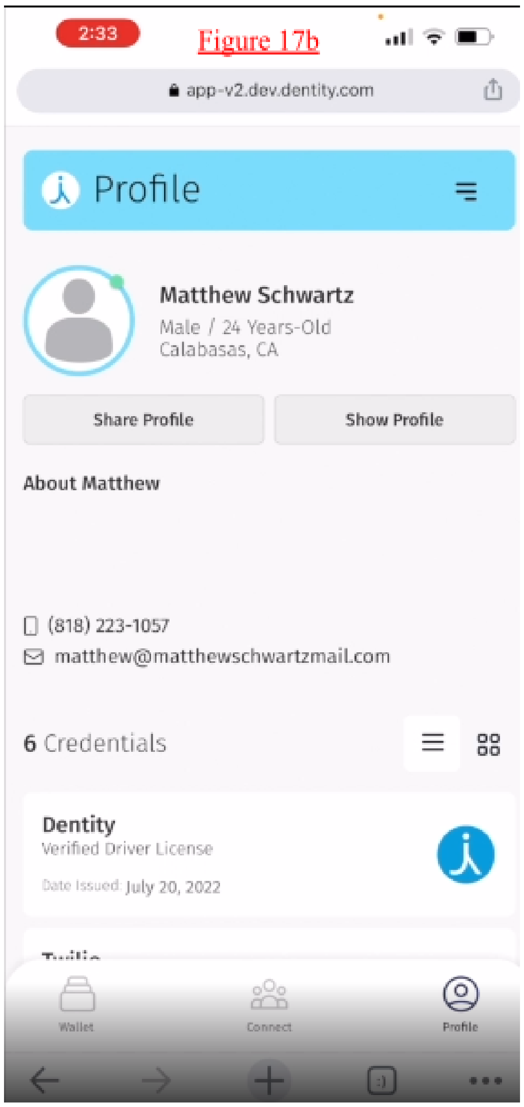
6 Credentials

John's Sandwich Shop
VIP Customer
Date Issued: August 1, 2022

Matthew Schwartz
VIP Crew
Date Issued: August 1, 2022

Wallet Connect Profile

Figure 17b



2:33

app-v2.dev.dentity.com

Profile

Matthew Schwartz
Male / 24 Years-Old
Calabasas, CA

Share Profile | Show Profile

About Matthew

(818) 223-1057
matthew@matthewschwartzmail.com

6 Credentials

Dentity
Verified Driver License
Date Issued: July 20, 2022

Wallet Connect Profile

mentioned about Trinsic Dentity is the process and how and where information is stored. Although Trinsic Dentity is fairly new, it provides a valid weakness of possible security concerns of how it is incorporated on the blockchain and what user data is still available for them to use or store within their databases. Unlike Microsoft which uses Azure for its Active Directory which is commonly known to be relatively safe in terms of proper encryption, Trinsic Dentity doesn't provide the "underhood" of how their DID software works. This can cause concerns because it can be somewhat easy to influence fraudulent activity or cause bad actors to potentially attack their own servers if sensitive information is on there. As this software is still growing and still providing ways how they can improve easier solutions to DIDs to make it seamless such as transactions through cryptocurrencies, it can become more prestigious where it can influence other systems and DID research software and even commercial software to better and upgrade their products as identification becomes more decentralized. As you can see, DID software is expanding with the new technologies being introduced in Web3.0 and more ideas are being spread on how to make identification easier. Through my research, I will explain some of the technical challenges that can be researched or looked at in the future to ultimately better the use of DIDs in the last segment of this report.

5 - Technical Challenges and Conclusion

In this last segment, I will be discussing a couple challenges that should be known about DIDs and DID software that can cause issues in the future. According to Fractal [Reference 12], an organization based on pushing for DID solutions, the biggest two challenges of DIDs are standardization and interoperability (Fractal, Tackling Challenges Section). Standardization is one of the most challenging break-through points for new ideas that begin to roll out and DIDs are one of them. DIDs must be able to gain the trust and be accepted by multiple organizations, governments and other ecosystems. This is really important because if big corporations do not accept DIDs, more people will eventually not use this new idea and it will eventually fail. For instance, if the government were to adopt IDs, it can make it a lot easier to present driver's licenses to the police for identification, and be able to use passports as DIDs to be accepted by airlines so you don't need to bring the physical document and risk losing it. The main reason why DIDs are being held back so heavily is due to the standardization process. For instance, what is the correct and viable way to create these standards of DIDs and how do we create and establish “grounds” so that DIDs are used under certain moderation? All these questions pose a challenge that can affect the future and usability of DIDs. Adopting a new idea can always be tough especially when it deals with identification, but standardization is another challenge within its own set of branching ideas of how it can be incorporated in our existing systems. Interoperability is another huge challenge that DIDs face due to the fact that it needs to incorporate a way to achieve communication across a data registry or a blockchain. In the beginning segment of this report, I have discussed this through W3C’s [Reference 2] example of a verifiable data registry. However, this data registry is simply some type of ledger, there needs to be a way to communicate across multiple platforms. Recall that Dock Certs and Trinsic Dentity use their own databases and separate blockchains for different reasons which means they are not interoperable with each other. This is a problem because then users will need to issue DIDs from different platforms and might even have to have separate wallets to accept different types of DIDs for each specific blockchain. One example I mentioned previously was with a university system creating degree DIDs for students when they have graduated. The university system will then have to find a common software DID that is commercial, not too expensive, and is able to transfer and exchange DIDs across multiple platforms and wallets. In a more simple example, if a user only has Zell and another peer has Venmo, it will be very annoying for these two to exchange money or perform transactions unless they both use a common blockchain for DIDs so they are interoperable. Currently, this issue is starting to be more secure as more companies such as Microsoft Entra are slowly adopting a common blockchain and influencing other companies to use. Of course, there are plenty more issues such as spending and cost, however, a lot of the issues with DIDs today are slowly diminishing as more organizations are finding solutions to tackle these problems. In the future, I believe that these challenges along with smaller problems will slowly decrease and we will see DIDs starting to be fully implemented in major corporations and it is only a matter of time until it does.

To conclude, I talked a lot about DIDs and what DIDs are and how they work along with their five major goals. I structured this report to break DIDs down into a more high-level understanding with plenty of examples and went on to discuss some of the more challenging aspects of DIDs and how they actually operate internally by discussing the architectural aspects, along with its identifier and syntax structure. I have also discussed the benefits of DIDs and why they can solve most of the common issues with our current identification system. I also broke down some downsides of DIDs and why they might not be a good choice to implement without projecting all aspects of this new identification system. I reflected this understanding by talking about 3 common research and commercial software: Dock Certs, Microsoft Entra, and Trinsic Dentity, and how they operate and use DIDs to verify and issue credentials. I have talked about the very good features of each including how they are experimenting to develop and integrate more simplistic methods to better their software. Not only that, I gave some weaknesses of each and explained how they can improve upon the creation of DID software. Lastly, I gave some technical challenges that can be thought about and researched in the future to ultimately better DIDs and fully incorporate them into a new identification system that will be more beneficial, accessible, and applicable to our society.

6 - References

1 - Valkama, Paul. "100,000 Years of Identity Verification: An Infographic History." *Trulioo*, 10 Nov. 2022, https://www.trulioo.com/blog/identity-verification/infographic-the-history-of-id-verification.

2 - "Decentralized Identifiers (Dids) v1.0." *W3C*, https://www.w3.org/TR/did-core/.

3 - Denenberg, Ray. "Uri Resource Pages." *About URIs: URI Resource Pages - IFLA CDNL Alliance for Digital Standards (ICABS) (Standards - Library of Congress)*, https://www.loc.gov/standards/uri/about.html#what.

4 - Seth, Shobhit. "What Is a Cryptocurrency Public Ledger, How It Works, Risks." *Investopedia*, Investopedia, 3 Nov. 2022, https://www.investopedia.com/tech/what-cryptocurrency-public-ledger/#:~:text=A%20cryptocurrency%20public%20ledger%20is,transactions%20executed%20between%20network%20participants.

5 - Pungtas, Tyron. "Did-URL Syntax." *Tyronzil*, https://www.tyronzil.com/scheme/did-url-syntax/.

6 - "Decentralized Identifiers (Dids)." *Decentralized Identifiers*, https://iherman.github.io/did-talks/talks/2020-Fintech/#/.

7 - Johnson-Ubah, Amarachi. "Benefits of Decentralized Identity." *Medium*, Veramo, 15 Sept. 2022, https://medium.com/veramo/benefits-of-decentralized-identity-df2cc1d7e1da#:~:text=Benefits%20of%20decentralized%20identity%20for,certificates%2C%20or%20documents%20are%20valid.

8 - "Decentralized Identity: The Ultimate Guide 2023." *Dock*, https://www.dock.io/post/decentralized-identity#how-do-you-create-decentralized-identifiers.

9 - Imi. "Decentralized Identity Management Risks - How SSI and Did Work." *Identity Management Institute®*, IMI Https://Www.identitymanagementinstitute.org/App/Uploads/2021/03/Logo-.Jpg, 28 Dec. 2022, https://identitymanagementinstitute.org/decentralized-identity-management-risks/#:~:text=Another%20potential%20issue%20of%20decentralized,may%20be%20revoked%20in%20others.

10 - Barclayn. “Tutorial - Issue Microsoft Entra Verified ID Credentials from an Application - Microsoft Entra.” *Tutorial - Issue Microsoft Entra Verified ID Credentials from an Application - Microsoft Entra | Microsoft Learn*, https://learn.microsoft.com/en-us/azure/active-directory/verifiable-credentials/verifiable-credentials-configure-issuer.

11 - “Trinsic Customer Story: Dentity.” *Trinsic*, 26 Oct. 2022, https://trinsic.id/customer-story-dentity/.

12 - Fractal. “Decentralized Identity Challenges & How to Tackle Them.” *Medium*, Fractal, 13 Oct. 2022, https://medium.com/frctls/decentralized-identity-challenges-how-to-tackle-them-7bf1f36e5f2c.