Ver. 2020

블록체인 기반의 안전한 인증 및 데이터 주권을 위한

W3C 탈중앙 ID 표준해설서

TTA 한국정보통신기술협회
Telecommunications Technology Association

머리말

2020년 신종 코로나바이러스감염증-19(COVID-19) 사태로 언택트(비대면) 기술 확산과 함께 산업의 중심이 오프라인에서 온라인으로 옮겨지면서 ICT의 중요성이 더욱 커지고 있습니다. 정부는 포스트 코로나 시대에 대응하여 '한국판 뉴딜' 정책을 발표로 '디지털 뉴딜', '그린 뉴딜', '안정망 강화'의 대형 프로젝트를 진행하여, D.N.A 생태계 강화, 비대면 사업육성, SOC 디지털화 등 코로나 이후의 위기를 극복하기 위해 디지털 국가로의 발전 정책을 추진 중에 있습니다.

D.N.A(Data-Network-AI)를 기반으로 디지털 국가와 스마트한 정부, 국토, 산업을 목표로 더 안전하고 편리해진 국민들의 삶 조성을 위하여 데이터댐, AI, 스마트의료, 디지털트윈(Digital Twin) 등의 ICT 기반 기술 발전과제를 선정하였습니다. ICT 표준은 R&D와 시장을 연결하는 핵심수단으로, ICT 분야의 기술적 상용화와 사용자 수요에 맞는 기술 개발을 위한 ICT 표준화를 추진하고 있습니다.

한국정보통신기술협회는 ICT 국제표준화전문가 활동지원을 통해 국내 산업체의 ICT 시장 경쟁력을 제고하고 국제표준화기구에서의 한국 영향력 확대를 위해 노력하고 있습니다. ICT 표준화포럼에서는 국내 산업체 등 중소기업의 수요를 반영하여 IETF의 I2NSF(Interface to Network Security Functions), W3C의 RTC(Real-time Communication Between Browsers), DID(Decentralized ID), Vehicle cloud(Vehicle Information Service Specification)의 최신 표준 4개를 선정하고, 관련 표준의 해설을 담은 ICT 국제표준화전문가 표준해설서를 마련하였습니다.

본 표준해설서는 해당 기술의 구현 및 지식습득이 필요한 국내 산업체와 ICT 국제표준전문가 표준화 활동에 교과서적인 전문적 지식전달 목적으로 활용될 수 있을 뿐만 아니라, ICT 사실표준화의 국내 산업 활성화에 기여할 수 있을 것으로 기대합니다. 마지막으로 표준해설서 발간에 참여해주신 사실표준화기구 미러포럼 표준전문가 여러분과 중소·중견기업 관련자 분께 깊이 감사드립니다.

목차

목차

1. DID(Decentralized ID, 탈중앙 ID) 표준 개요

1) W3C에서의 탈중앙 운동

DID(Decentralized ID, 탈중앙 ID) 표준 등장의 기술적 맥락을 이해하기 위해서는 W3C(World Wide Web Consortium) 내의 탈중앙 운동에 대한 이해가 선행되어야 한다.

W3C 내에서의 탈중앙 움직임은 웹을 창시한 팀 버너스리(Sir Tim Berners-Lee)로부터 시작되었다. 1989년 W3C를 창시하고 HTML을 표준화하기 시작했던 팀 버너스리의 본질적 목적은 웹을 상호 공유하고 이를 통해 자유롭게 데이터를 사용자들끼리 주고받는 것이었다.

하지만 2000년대 이후로 데이터 주권이 사용자들이 아닌 기업에 넘어가기 시작하면서 데이터는 특정 기업에 종속되는 이른바 중앙화(Centralized)되기 시작했다. 대부분의 소셜 데이터는 페이스북이 대부분 차지하게 되었고, 사용자 광고 정보는 구글이, 동영상과 관련한 콘텐츠는 유튜브가 대다수의 데이터를 보유하게 되었고, 이러한 데이터는 사용자들이 만든 데이터임에도 불구하고 사용자는 해당 데이터를 서비스 제공자의 결정에 따라 접근할 수도 있고 접근하지 못할 수도 있는 상황이 된 것이다.

데이터 주권이 사용자가 아닌 서비스 기업에 종속되는 이러한 웹 생태계에 대한 불만이 지속적으로 W3C 내에 대두되었고, 팀버너스리와 W3C 스탭들은 이러한 웹 생태계의 불공정성을 바로잡고자 탈중앙(Re-Decentralized) 운동을 전개하기 시작한다.

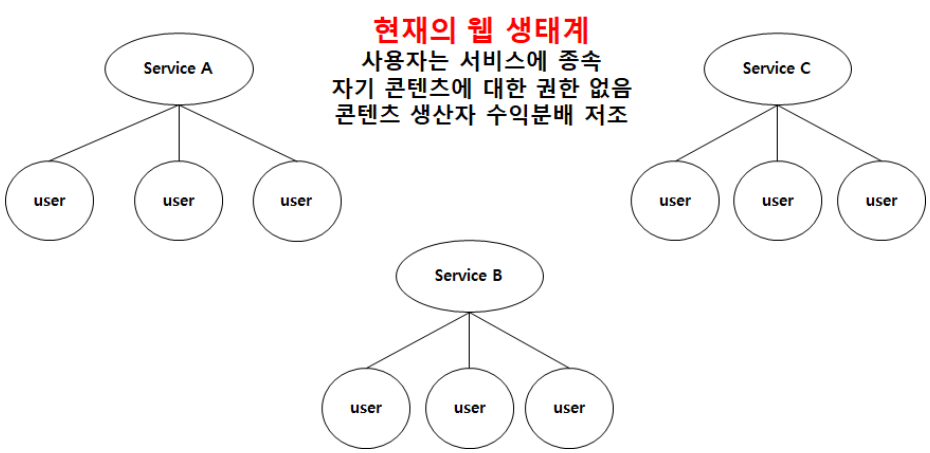

그림 1. 특정 기업에 종속되어 있는 현재의 웹 생태계

W3C에서 진행되고 있는 탈중앙 운동에서 가장 중요한 부분은 데이터의 연동이다. 각각의 서비스 제공 기업이 독점적으로 데이터를 보유하게 된 것은 타 서비스와의 연동이 불가능하기 때문이다, 이에 먼저 탈중앙 생태계를 향해 나가기 위해서는 서비스 간의 데이터 연동이 가장 중요한 사항으로 대두되기 시작했다.

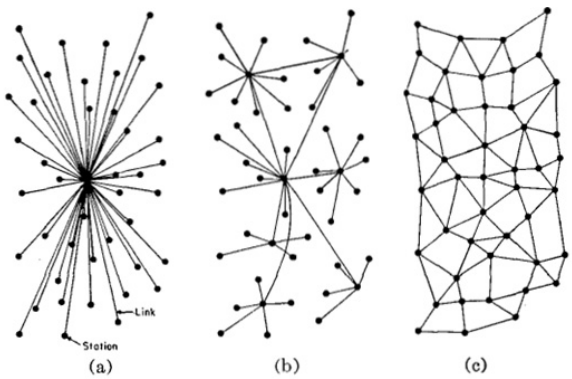

Fig. 1—(a) Centralized. (b) Decentralized. (c) Distributed networks.

그림 2. 팀 버너스리를 중심으로한 탈중앙 웹 운동의 진행

2) 탈중앙을 위한 주요 표준

데이터 연동을 위해서 선행되어야 하는 것은 해당 데이터가 어떤 종류의 데이터인가를 결정하는 시맨틱 웹 관련 부분이다. 왜냐하면, 각각의 서비스가 연동하기 위해서는 특정 데이터가 타 서비스에서도 매핑되어야 하는데, 매핑을 하기 위해서는 해당 데이터가 어떤 종류의 데이터인가에 대한 의미도 함께 전달되어야 하기 때문이다.

W3C에서는 데이터 연동의 근간이 되는 JSON-LD와 소셜 서비스 간의 데이터 연동을 위한 소셜웹 표준을 본격적으로 진행하게 되었다.

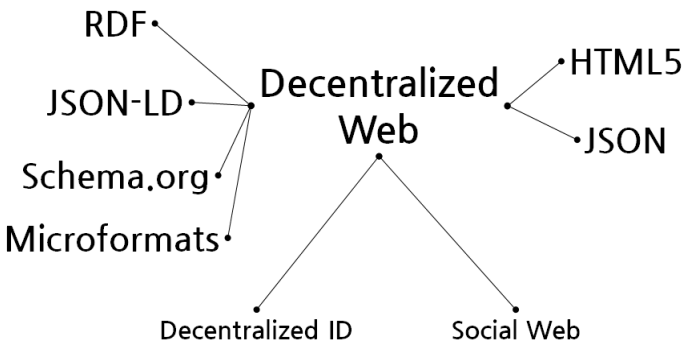

그림 3. 주요 탈중앙 웹 표준 현황

(1) JSON-LD

JSON-LD는 JSON(Javascript Object Notation) 기반의 Linked Data의 약자로서 서로 다른 이기종 서비스 간에 데이터 교환 시 해당 데이터에 대한 의미를 함께 전달하여 각각의 서비스 제공자 사이의 데이터 연동을 가능케 하는 기본적인 표준이다.

아래 예제 샘플에서 확인할 수 있듯이 기본적인 JSON 기반으로 데이터를 전달하지만, @ 키워드를 통해 해당 데이터가 어떤 종류의 데이터인지 사전적 의미도 URL을 통해 함께 전달한다.

표 1. JSON LD 샘플 예제

```
{
  "@context": “http://schema.org/person",
  "@id": "http://dbpedia.org/resource/John_Lennon",
  "name": "John Lennon",
  "born": "1940-10-09",
  “email": “kim@naver.com”,
  “phone": "(540) 961-4469",

}
```

(2) 소셜웹 표준

W3C의 소셜 서비스 표준 중 소셜 활동 용어(Activity Vocabulary)와 소셜 활동 흐름(Activity Stream) 두 개의 표준은 기존에 오픈소셜재단(Open Social Foundation)에서 개발된 내용을 상당 부분 차용하였다.

W3C 소셜 웹 워킹 그룹 표준 중 연동 가능한 소셜 서비스를 구현하기 위한 기본이 되는 실제적인 표준은 액티비티펍(ActivityPub)이다.

액티비티펍은 먼저 사용자를 액터(Actor)로 정의한다. 이 액터가 소셜 서비스 데이터의 수신함(InBox)과 발신함(OutBox)을 보유하고 있으면서 보유된 데이터를 자신이 원하는 서버에 전송한다. 여기서 소셜 표준을 준수하고 있는 서버는 어디라도 해당 데이터를 저장할 수 있고, 서버와 서버 사이에서도 데이터 연동이 가능하다.

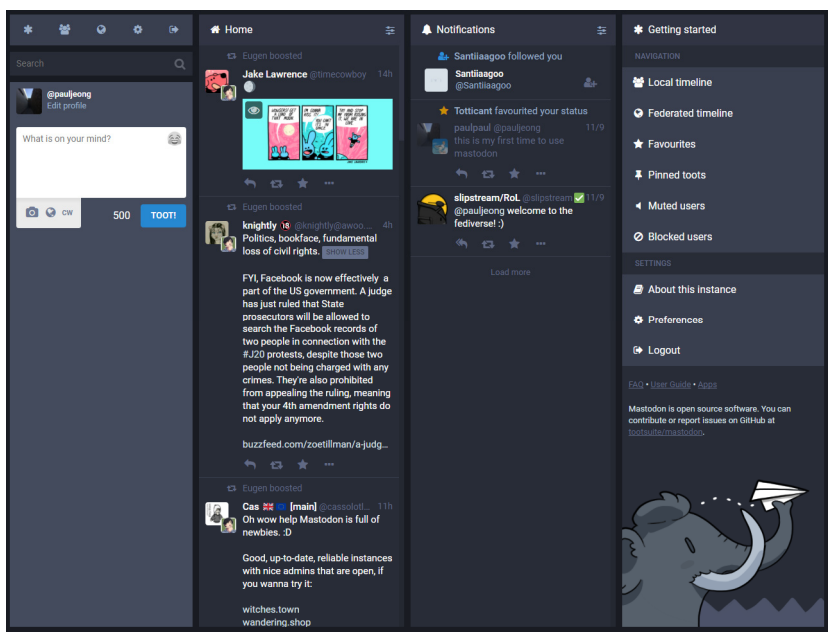

그림 4. W3C 소셜 웹 표준으로 개발된 mascodon 서비스

전술한 바와 같이 W3C 소셜 웹 표준은 탈중심 개념을 기반으로 하여 현재 10개의 소셜 표준이 공표되었다.

현재 공표된 표준은 아래와 같다.

표준명	설명	비고
소셜 활동 흐름 (Activity Streams) 2.0	소셜 서비스 내에서 서버-서버, 클라이언트-서버 간의 소셜 활동에 대한 구문과 맥락을 정의하기 위한 표준	Recommendation
소셜 활동 용어 (Activity Vocabulary)	소셜 서비스 내에서 가능한 활동들에 대한 용어 정의와 해당 활동 개체에 필요한 요소들을 정의한 표준	Recommendation
마이크로펍 (MicroPub)	마이크로포맷 기반의 간단한 소셜 데이터 전송과 연동을 위한 표준	Recommendation
연결된 데이터 알림 (Linked Data Notifications)	알림 데이터 내에 포함되어 있는 콘텐츠를 링크 기반으로 제공하여 해당 콘텐츠가 재사용될 수 있도록 하는 표준	Recommendation
웹멘션 (Webmention)	특성 소셜 서비스 내에서 링크된 누군가의 리소스가 언급되었을 때 언급되었음을 원천 사용자에게 알려줄 수 있도록 하기 위한 표준	Recommendation
액티비티펍 (ActivityPub)	탈중심 소셜 서비스를 구현하기 위한 기본적인 스펙과 내용을 담은 표준으로 소셜 웹 서비스의 가장 상세한 스펙을 담고 있음.	Recommendation
웹섭 (WebSub)	소셜 서비스 상에서 구독(Subscription)을 지원하기 위한 표준	Recommendation
포스트타입발견 (Post Type Discovery)	소셜 서비스 내에서 포스팅되는 데이터의 타입이 무엇인지 판별하기 위한 알고리즘 표준	Recommendation
소셜웹규약 (Social Web Protocols)	소셜 웹 워킹 그룹에서 개발 중인 각각의 표준에 대한 해설	Recommendation
제이슨포맷포스직렬포맷 (JF2 Post Serialization Format)	소셜 서비스 내에서 싱글 데이터를 포스팅하기 위해 사용되는 제이스 직렬 포맷 표준	Recommendation

3) 블록체인 기술과 W3C DID 표준의 시작

(1) 블록체인 기술의 대두

비트코인 열풍으로 더 잘 알려진 블록체인은 데이터를 블록 기반으로 저장하되, 각각의 데이터가 해시 기반으로 연결되도록 하는 기술이다.

예를 들면 블록 0에 기본적인 데이터를 저장하고 해당 데이터를 암호화한 해시 값을 블록 1에 저장한다. 블록 1의 데이터는 역시 암호화하여 해당 해시 값을 블록 2에 저장한다.

이렇게 데이터가 저장될 경우, 블록 0의 데이터를 변조할 경우, 해당 데이터를 해시한 값이 블록 1에 있는 해시값과 달라지기 때문에 블록 0의 데이터가 변조되었다는 것을 확인 할 수 있다.

결국 블록 0의 데이터를 조작하려면 블록 2의 데이터를 먼저 조작하고 조작한 값으로 다시 블록 1의 데이터를 조작해야 하며 블록1의 데이터까지 조작을 해야 블록 0의 데이터를 완벽하게 조작할 수 있게 된다.

이러한 블록이 한두 개가 아니라 수십 개, 수만 개에 이를 경우, 특정 블록의 데이터를 조작하려면 수만 개의 블록을 차례차례 조작해야 하기 때문에 물리적으로 블록 데이터를 조작하기는 불가능하다.

뿐만 아니라 하나의 블록체인만 존재하는 것이 아니라 이러한 블록체인 데이터 장부가 각각의 노드만큼 존재하기 때문에 한 블록체인의 데이터가 조작되었다 하더라도, 다른 블록체인 장부와 비교할 수 있기 때문에 해당 데이터 조작여부를 확인할 수 있다.

이와 같이 각각의 데이터가 암호화된 값으로 연결되어 있고, 이러한 장부가 상당수 존재하기 때문에 블록체인 내 특정 값을 변조하는 것은 거의 불가능한 것이다.

블록체인의 구현 방식은 크게 두 가지로 나뉠 수 있는데 private 블록체인과 public 블록체인이 그것이다.

부분	private 블록체인	public 블록체인
접근성	특정인만 접근 가능	누구나 접근 가능
장부 수	소수	다수
장점	특정 네트워크 내에서 구동되어 기관 등의 독립 구성에 적당	장부 수가 많아서 조작이 불가능
단점	블록체인 자체의 조작 가능성 상승	블록체인 장부별 합의 알고리즘 등의 복잡성 상승

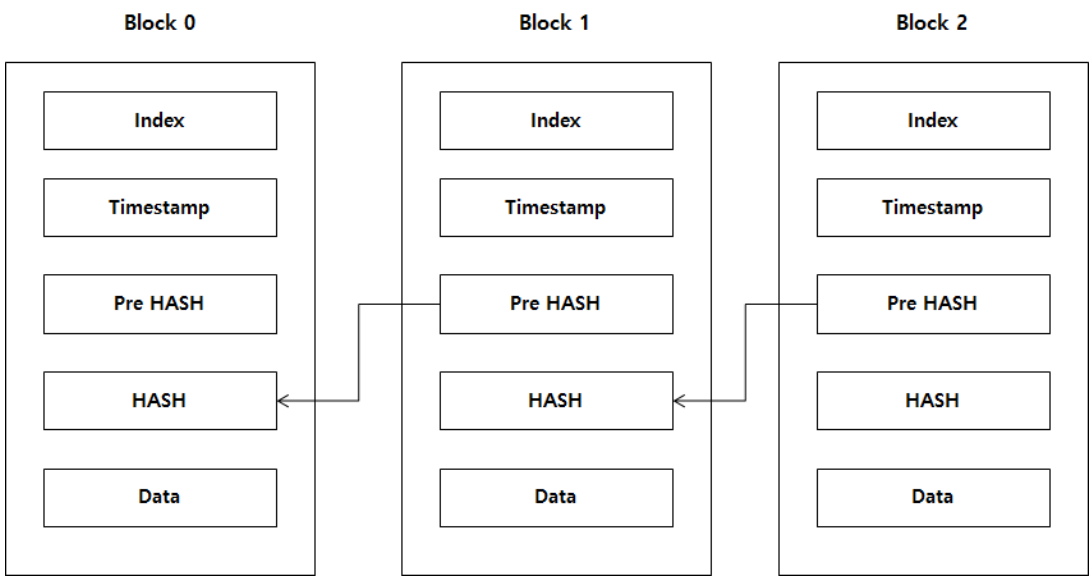

그림 5. 블록체인 기술 개요

(2) 블록체인 기술과 웹의 만남

① 블록체인 기술과 웹의 만남

2016년에 블록체인 워크샵을 통해 블록체인 기반의 ID를 표준화하자는 의견이 개진되었고, 2018년이 되어서야 W3C에서 DID 워킹 그룹이 공식적으로 결성되었다.

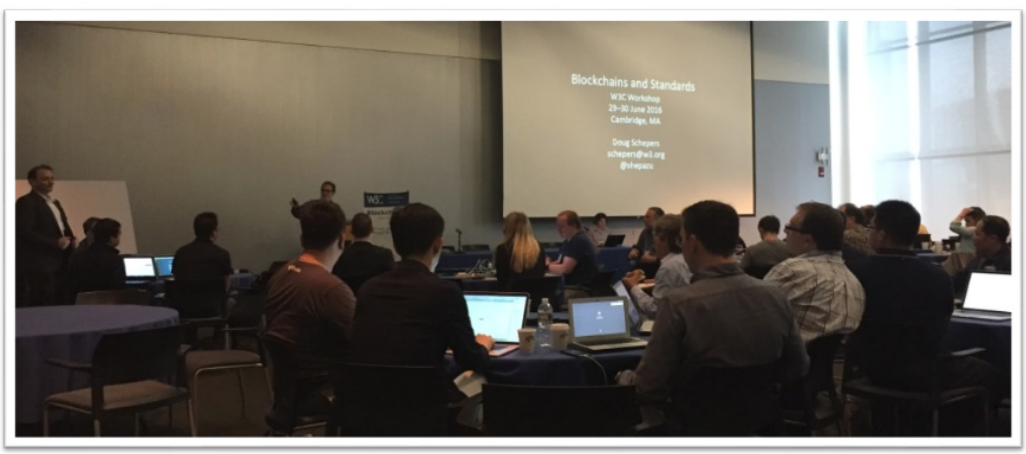

그림 6. 2016년에 MIT에서 개최된 W3C 블록체인 워크샵

2016년에 MIT에서 개최된 블록체인 워크샵은 블록체인이 어떻게 웹과 융합될 수 있는지, 블록체인 자체에 대해서 W3C에서 표준화할 수 있는 항목은 어떤 것들이 있는지에 대해 상당히 다양한 논의가 개진되었다.

이러한 논의를 통해 워크샵은 3개의 주요 의제를 마련했는데 아래의 3가지로 요약할 수 있다.

구분	내용	비고
IDENTITY	블록체인 기반으로 기존의 인증방식을 어떻게 개선할 수 있는지에 대한 논의	DID 워킹 그룹으로 생성
Provenance	관리되는 자원 등의 리소스에 대한 이력 등을 어떻게 관리할 수 있는지 논의	
API / Platform	사물인터넷 등의 플랫폼은 혹은 블록체인 기반 플랫폼 간의 API는 어떻게 구현할 수 있는지 논의	

이 워크샵 이후 기본적인 표준 초안을 만들고 W3C에 정식으로 DID 워킹 그룹 제안을 하게 된다. 워킹 그룹 결성은 JSON-LD 개발을 주도한 Manu Sporny와 Drummond Reed가 주요 멤버로 참여하였으며 2021년까지 표준개발 완료를 목표로 현재 활발하게 진행 중이다.

	1세대 인증	2세대 인증	3세대 인증
방식	중앙화	하이브리드	탈중앙화
내용	서비스 제공자가 아이디 비밀번호 기반으로 저장	제삼자가 OAuth 기반으로 제공	블록체인 기반으로 사용자가 소유
인증서 보관장소	서비스 서버	제삼자 서버	사용자가 지정
예시	일반 웹사이트	페이스북 / 카카오톡 로그인	개발중
장점	단순	일괄 관리 가능	자기 주권 해킹등의 대량 정보 유출 없음
단점	해킹 등의 정보 유출	해킹 등의 정보유출 특정 플랫폼에 종속	초기 설정 어려움

그림 7. 인증 방식의 변화

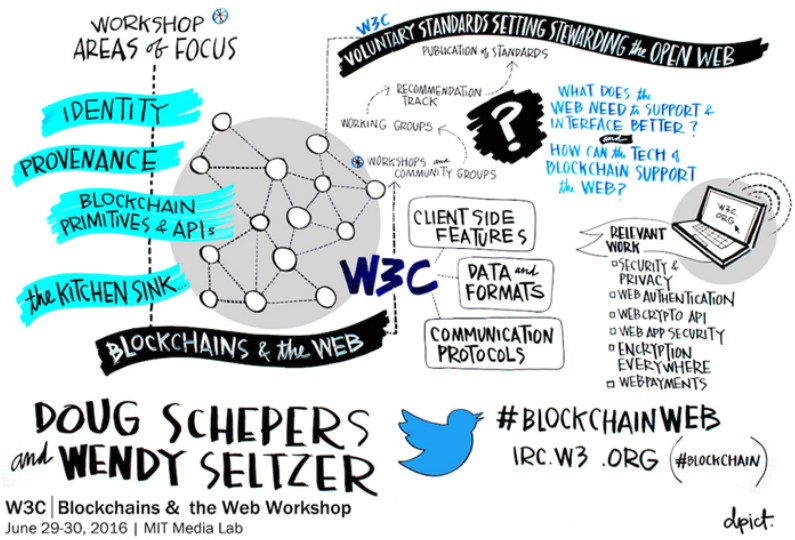

그림 8. W3C 블록체인 워크샵을 통해 진행된 브레인스토밍

4) DID 주요 개념

DID(Decentralized ID)는 블록체인 기반으로 사용자 정보를 저장하도록 하여 안전하고 편리한 인증이 가능하도록 하고, 인증 정보를 제 3자에게 맡기는 것이 아니라 사용자가 직접 관리하도록 하여 데이터 주권을 사용자에게 가져다 주는 SSI(Self Soverign Identity)의 일종이다.

(1) DID의 유즈케이스와 이에 따른 개념적 객체

DID 표준의 개념적 주요 객체는 DID 주체, 인증 및 발급기관, 검증기관, DID 저장소로 나뉠 수 있다. DID 주체와 인증 및 발급기관은 별도의 DID 문서를 기반으로 DID 저장소에 등록되고, DID 주체의 요구에 따라 인증 및 발급기관이 DID 주체의 증명서를 발급하면 DID 주체가 이를 검증기관에 제출하고 검증기관은 블록체인에 보관 되어있는 DID 문서의 공개 키 등을 이용하여 해당 증명서의 진위여부를 증명한다.

예를 들어, 한 사용자(Holder)가 자신의 졸업 증명서를 기업(Verifier)에 제공하고자 할 때, 그 사용자는 대학교(Issuer)에 증명서를 요구하고, 인증기관은 개인 키(Private Key)와 함께 해당 증명서를 사용자에게 제공한다. 이 증명서를 Claim이라고 하는데, 사용자는 해당 claim 중 졸업에 해당하는 부분을 verifier에 해당하는 기업에 제출한다. 기업은 해당 증명이 실제 유효한지 Claim에 포함되어 있는 개인 키와 블록체인에 저장 되어있는 공개 키를 이용하여 문서를 검증한다.

① DID 주체(Holder)

DID 주체는 DID 가 지칭하는 실제 객체를 의미한다. 사람이 될 수도 있고, 특정 기관, 물건, 사물 등이 여기에 속한다. DID 주체는 DID 문서를 작성하여 해당 DID 문서를 블록체인에 저장한다. DID 문서에는 DID 주체가 생성한 공개 키를 포함하고 있다.

② 인증 및 발급기관(Issuer)

인증 및 발급기관은 DID 주체가 요구하는 증명서를 발급하는 기관을 의미한다. 이 발급기관 역시 공개 키가 포함된 DID 문서를 생성하여 사전에 블록체인에 저장해야 한다.

③ 검증기관(Verifier)

DID 주체가 제출하는 증명서를 검증하는 기관을 의미한다. 기업이나 은행 등 사용자가 제출하는 증명서의 진위 여부가 중요한 기관으로, 제출된 증명서가 문제가 없는지 증명서 내에 포함되어 있는 개인 키와 DID를 이용해 DID 문서를 저장소에서 추출(resolve)하고 DID 문서 내의 공개 키를 이용해 증명서의 진위 여부를 검증한다.

④ DID 저장소(블록체인)

DID 저장소는 블록체인 기반으로 DID 문서를 저장하는 저장소이다. 블록체인 기반으로 구성되어 있어 위조나 변조가 불가능하다.

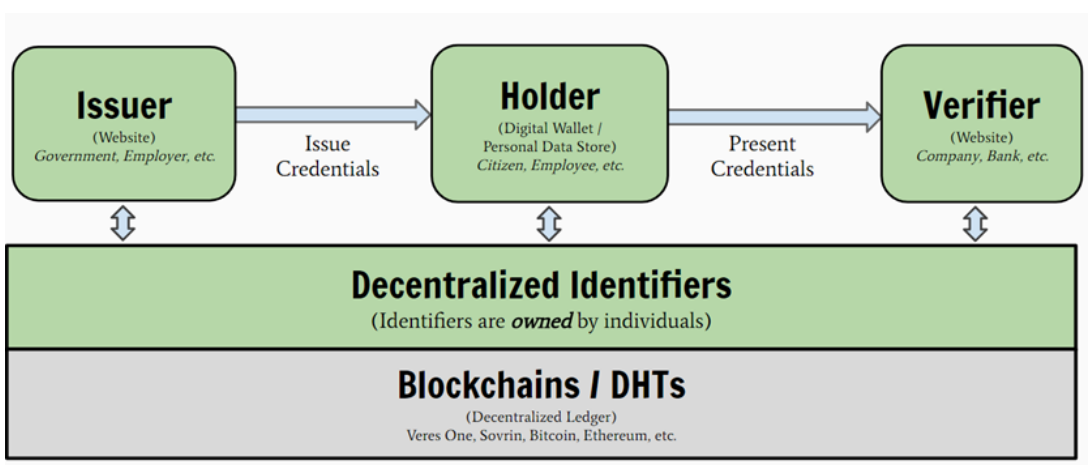

그림 9. 기본 DID 개념도

(2) DID 아키텍처 및 이에 따른 주요 구성 요소

DID는 DID 주체(DID subject), DID 컨트롤러(DID controller), DID, DID 문서(DID document), DID 메소드(DID Method), DID 분해자(DID resolver), 검증 가능한 데이터 저장소(Verifiable Data Registry) 등 모두 7개의 주요 요소로 구성되어 있다.

DID 주체는 DID 식별자가 가리키는 주체 그 자체를 의미한다. 사람이 될 수도 있고, 기관, 사물 등 식별에 대한 실제 객체라고 할 수 있다.

DID 컨트롤러는 DID 주체가 DID 문서를 생성하도록 하는 기능을 의미한다. 컨트롤러는 DID 주체 자체일 수도 있고, DID 주체의 위임을 받아 처리하는 별도의 기관일 수도 있다.

DID는 DID 주체를 식별하기 위한 식별자로 일련의 문자열로 구성된다. 관련 내용은 4. 가. 식별자에서 자세히 설명한다.

DID 메소드는 DID를 생성하는 일련의 알고리즘으로 데이터 저장소의 종류에 따라 달라질 수 있다.

DID 문서는 생성된 DID의 공개 키와 일련의 DID 주체와 관련된 메타 데이터로 구성된 문서로서 실제 블록체인 등의 데이터 저장소에 보관되는 문서를 의미한다.

DID 분해자는 주어진 DID가 실제로 데이터 저장소에 보관되어 있는지 해당 DID로 조회하여 DID 문서를 반환하도록 하는 역할을 수행한다.

검증 가능한 데이터 저장소는 블록체인 기반으로 DID 문서를 보관하는 저장소를 의미한다.

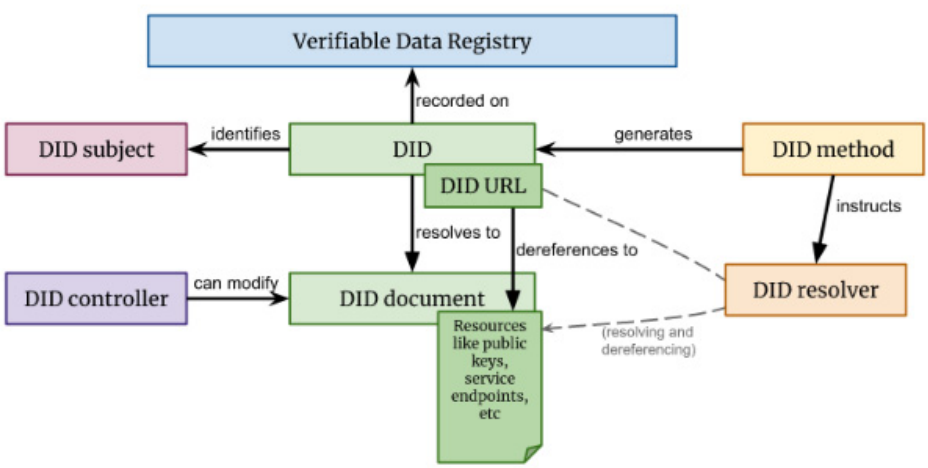

그림 10. DID 표준의 주요 구성요소

2. W3C 및 DID 워킹 그룹

1) W3C 소개

(1) W3C 설립 배경

1989년 스위스의 CERN 연구소에서 팀 버너스리가 컴퓨터 간의 문서를 교환하기 위한 표준인 HTML을 발명하였다. 이후 해당 표준을 지속적으로 발전시키기 위해 팀 버너스리는 HTML 기술을 오픈소스로 공개하고 1994년 10월, 기업, 공공기관, 대학을 중심으로 W3C(World Wide Web Consorthium, W3C)를 결성하였다.

(2) 주요 조직 구성

W3C 조직은 두 가지 형태로 이루어져 있다. 먼저 표준 개발 및 웹 기술 전반을 관활하는 Director 중심의 조직과 W3C 운영 및 회원 관리, 재정 등을 관활하는 CEO 중심의 조직이다. Director의 경우 1994년 설립 이후로 팀 버너스리가 꾸준히 직책을 수행하고 있고, CEO는 Jeff Jaffe가 수행하고 있다.

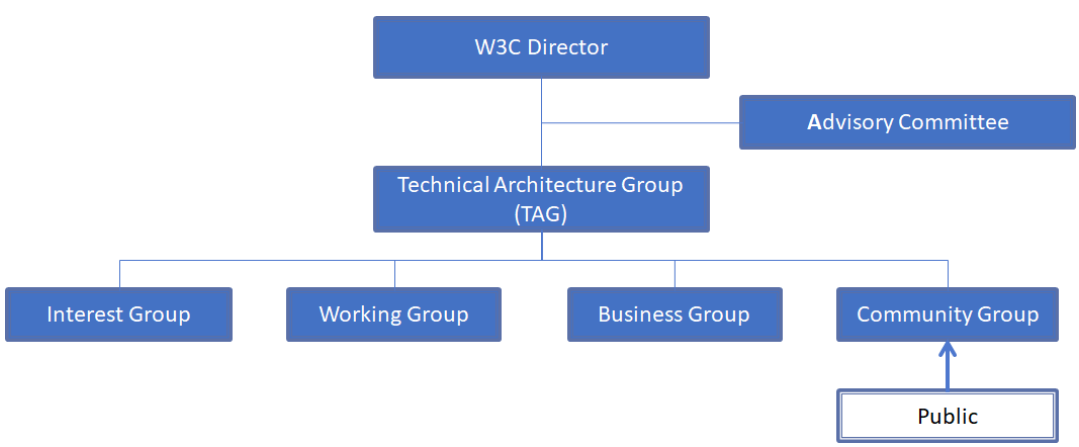

그림 11. W3C 표준 관련 조직도

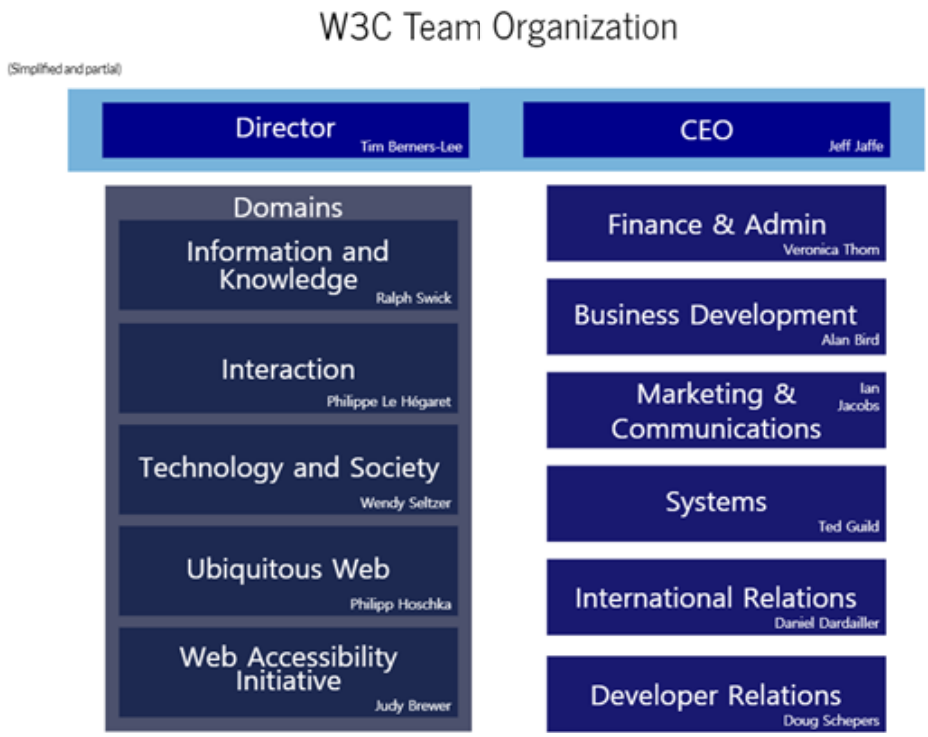

그림 12. W3C 조직 구조

(3) 주요 개발 표준 내용

W3C 설립 당시 W3C의 표준 개발 범위는 웹 브라우저에서 표현되는 마크업 관련 기술이었으나, 현재는 브라우저 표현 기술에 대한 표준뿐만 아니라 데이터 연동을 위한 서버-서버 간 연동 기술도 표준으로 개발하고 있다. 웹에 근간이 되는 가장 주요한 표준은 HTML, CSS이며 JSON-LD, 사물웹(Web of things) 등 연동과 관련한 표준도 다양하게 진행되고 있다.

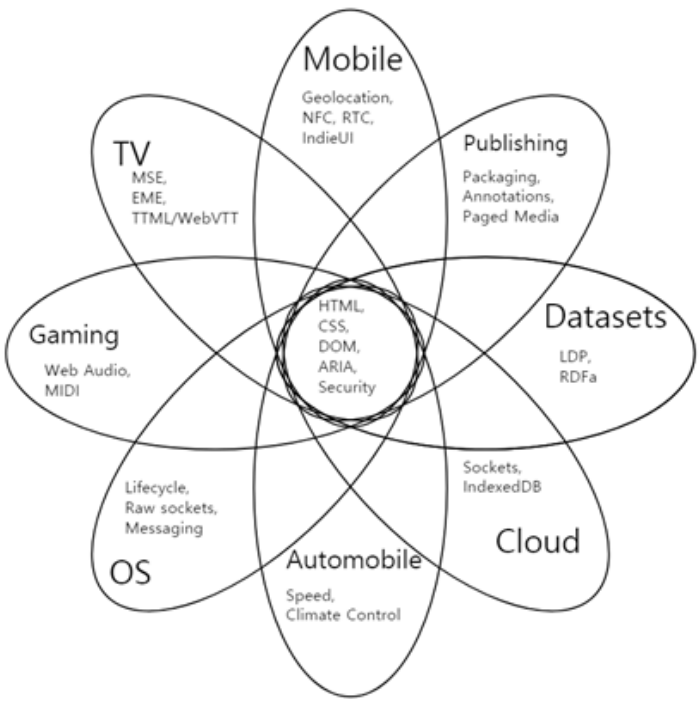

그림 13. W3C 표준 주요 내용

2) DID 워킹 그룹 소개

(1) 워킹 그룹 소개

DID(Decentralized Identifier) 워킹 그룹은 2014년 W3C 블록체인 워크샵을 기점으로 개인 식별자를 블록체인 기반으로 생성하고 활용하기 위해 2019년 9월에 결성되었다. 해당 워킹 그룹을 결성하는 데 주요한 인물로는 JSON-LD를 창안한 Manu Sporny와 Daniel Burnett 등이 참여하였으며 현재 3개의 주요 표준 문서를 개발하고 있다.

표 2. DID 워킹 그룹 구성

구분	내용	비고
시작일	2019년 9월	
종료 예상일	2021년 9월	
의장	Daniel Burnett Brent Zundel	
미팅	매주 1회	online

(2) 주요 표준 내용

현재 Decentralized Identifier Use cases & Requirements, Decentralized Characteristics Rubric 등 2개의 그룹 노트 문서 작업과 Deentralized Identifiers Data Model and Syntax 1개의 공식 표준 문서 등 총 3개의 문서가 개발 중이다.

표 3. DID 주요 표준 문서

표준명	문서 상태
Decentralized Identifier Use cases & Requirements	WD
Decentralized Characteristics Rubric	WD
Deentralized Identifiers Data Model and Syntax	WD

3. DID 표준 유즈케이스

1) DID 주요 서비스 모델

(1) 인증

DID 기술을 이용한 가장 대표적인 서비스 중에 하나로 사용자 인증을 블록체인 기반으로 진행할 수 있다. 기존의 아이디, 패스워드 기반의 인증 방식은 사용자 정보가 모두 특정 서비스 제공자의 서버에 보관되어 사용자도 모르게 정보가 유출될 수 있고, 서비스 제공자가 해당 정보를 타 업체에 넘겨 불합리한 수익을 창출할 수도 있다.

DID는 특정 서비스 제공자에 좌우되지 않고 자신의 정보는 자신이 원하는 장소에 개별적으로 보관할 수 있어, 대규모의 정보 유출은 불가능하다.

(2) 전자 증명서

기존의 증명서는 대부분 문서 형태로 발급되고 제출되는 형태였고, 사용자는 필요한 만큼 해당 문서를 비용을 지불하고 발급/인쇄하여 사용했고 위변조 논란도 있을 수 있었다. DID 기반의 전자 증명서는 문서를 직접 다루어야 하는 불편함을 해소하고 위변조 논란을 잠재울 수 있다. 특정 사용자가 자신의 성적 증명서를 발급받고자 출신 학교에 요구를 하면 출신 학교는 블록체인 등의 저장소에 자신의 DID를 발급하여 공개 키와 함께 저장하고 사용자에게 개인 키가 담긴 증명서를 제공한다. 사용자 역시 자신의 DID를 블록체인에 저장하고 전달받은 증명서에 자신의 개인 키를 담아 증명서를 요구하는 기관에 제출한다. 증명서를 받은 기관은 증명서 내의 개인 키와 블록체인 내의 공개 키를 활용하여 해당 문서의 진위를 판가름할 수 있다.

(3) 출입통제

일부 정부 기관과 기업에서 DID 기반의 출입통제 서비스는 시범 운용 수준으로 서비스되고 있다. 기업의 직원은 자신의 DID를 발급하여 특정 블록체인에 저장하고 출입시에 자신의 DID 번호가 담긴 스마트폰 등을 통해 출입이 가능하다. 그동안 출입자의 개인정보가 회사 등의 서버에 저장될 수 밖에 없었는데, DID 기반의 출입통제는 회사 서버에는 아무 정보도 보관하지 않고 DID를 받아서 블록체인 검증을 통해 해당 DID의 유효성 여부만 체크하면 되기 때문에 훨씬 안전하고, 별도의 중앙 서버가 필요 없어서 비용도 상당히 절감된다.

2) DID 시장 동향

(1) DID 얼라이언스 동향

DID 서비스 자체가 별도의 독립 기업이 서비스하기 보다는 다양한 기업이나 비영리 기관들과의 협업이 중요하기 때문에 국내외에서 다양한 얼라이언스가 설립되어 활동 중이다.

① 마이아이디 얼라이언스

마이아이디 얼라이언스는 아이콘루프가 주도하여 2019년 9월 경에 설립된 연합체로 아이콘루프가 개발한 분산 ID를 기본 플랫폼으로 한다. 마이아이디는 신한은행, 부산은행 등의 은행과 삼성증권, KB증권 등 증권사 등이 대거 참여하고 있다.

마이아이디 얼라이언스는 주로 금융권의 개좌 개설, 금융거래를 DID 방식으로 서비스하기 위한 목적이 크다.

② 이니셜 컨소시엄

이니셜 컨소시엄은 과학기술정보통신부와 한국인터넷진흥원이 주관하는 '2019 블록체인 민간주도 국민 프로젝트'를 수행하는 기업들 중심으로 결성되었다.

SK텔레콤, LG유플러스, KT 등의 통신사와 삼성전자 등이 참여해 2019년 7월에 결성되었으며 모바일 전자 증명과 관련한 서비스를 선보이는 것을 목적으로 하고 있다.

이니셜 컨소시엄은 SK텔레콤 중심으로 개발된 하이퍼레저를 기반으로 하여, 기존 증명서 발급 및 제출, 구매 확인서 등의 전자 증명서 서비스를 목표로 하고 있다.

③ DID 얼라이언스 코리아

DID 얼라이언스 코리아는 '금융결제원'과 '한국전자서명포럼', '한국 FIDO 산업포럼'이 주축이 되어 2019년 10월에 결성되었다. 주요한 목적은 DID 신원인증 인프라 구축에 필요한 글로벌 표준화 및 이를 시장에 확산하는 것이다.

DID 얼라이언스 코리아는 라온시큐어가 개발한 DID 옴니원을 기반으로 하고 있다.

(2) DID 서비스 및 시장 동향

DID 관련 서비스는 아직 시장에 안정적으로 서비스 되는 모델은 없으나 국내외에서 다양한 기업들이 서비스를 선보이고 있다.

마이크로소프트는 자사가 개발한 DID 시스템을 비트코인 네트워크에 활용하겠다고 발표하며 DID 서비스의 시작을 알렸다. 2020년 6월 마이크로소프트는 오픈소스 기반의 DID 솔루션 '아이온'의 퍼블릭 버전을 비트코인 메인 네트워크 위에 론칭했다고 밝혔는데, 추후 오픈소스 기반의 개방형 시스템으로 지속적으로 지원하기로 했다.

SKT와 삼성은 삼성의 블록체인 키스토어와 SKT의 모바일 전자증명 서비스인 이니셜을 연동해 모바일 디바이스 등을 통해 서비스를 제공하겠다고 밝혔다. SKT와 삼성전자는 이번 협업을 바탕으로 향후 DID 기반의 높은 보안 수준을 유지하면서 다양한 전자증명 서비스를 지속적으로 선보일 계획이며 이니셜 컨소시엄을 통해서도 관련 업계와의 협업도 진행하기로 했다.

4. DID 표준 해설서 주요 개념

1) 식별자(Identifier)

이 장에서는 DID 식별자 구문 자체에 대한 설명을 제공한다. DID는 아래와 같이 기본적인 형태로 구성된다.

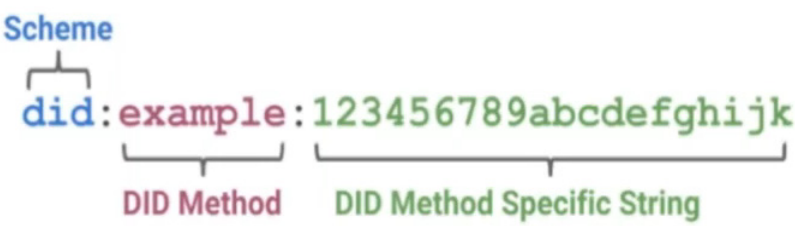

그림 11. DID 구문

did는 본 ID가 DID라는 것을 표현하기 위한 schema를 나타내고 두 번째는 메소드를 표현한다. 메소드는 해당 DID가 블록체인에 저장될 DID 문서와 매핑되는 알고리즘을 설명하는 것인데 현재 개발 되어있는 개별 블록체인의 단축 값으로 표현된다. 마지막으로 DID 메소드에 의해 생성된 고유한 값이 문자열이 마지막에 따라 붙는다.

2) 데이터 모델(Data Model)

이 장에서는 'DID 문서(DID Document)'에 포함되어야 할 데이터 모델에 관하여 설명한다. 'DID 문서'라 함은 DID와 관련한 메타 정보가 기록된 문서이다. DID 문서는 JSON LD 형태로 구성되어 DID 주체를 식별하는 DID, DID 문서를 생성하는 컨트롤러, 인증을 위한 공개 키, 선언 메소드 등으로 구성되어 있다. 이 DID 문서가 실제 블록체인 등의 저장소에 저장되는 것이다.

표 5. DID 문서 샘플

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

    "type": "VerifiableCredentialService",
    "serviceEndpoint": "https://example.com/vc/"
  }]
}
```

3) 핵심 속성(Core properties)

이 장에서는 DID 문서 내에 포함되어야 할 주요 속성에 대하여 설명한다. DID 문서 내에 포함되는 주요 속성은 다음과 같다.

id	DID 주체(DID subject)를 지시하기 위한 id
controller	DID 주체 대신에 DID 문서 등을 생성하는 역할을 하는 컨트롤러
verificationMethod	DID 주체나 DID를 이용하는 관련 당사자들이 문서의 검증을 위해 사용하는 검증 방법
authentication	DID 서비스 내에서 DID가 지시하는 주체가 맞는지 또는 DID 컨트롤러가 DID 주체를 대신해서 활동하는 것이 맞는지를 검증하는 방법
assertionMethod	DID 주체를 대신하였다는 사실을 검증하기 위한 방법
keyAgreement	DID 내의 특정 당사자가 DID 주체를 대신하여 문서 생성에 참여할 때 사용할 수 있는 관계 검증을 표현하는 데 사용
capabilityDelegation	DID 내의 특정 당사자가 DID 주체를 대신하여 기능을 호출할 때 관계검증을 하는 데 사용
capabilityInvocation	
service end point	DID 문서에서 DID 주체와 DID 내의 타 당사자와 통신하는 방법을 표현하는 데에 사용

4) 주요 표현 방식(Core representatives)

DID 문서에 대한 모든 표현은 본 표준 문서에서 정의된 데이터 모델에 따라 모호하지 않게 파싱되도록 정확한 매핑 과정에 따라 직렬화되어야 한다. 본 섹션에서는 이와 같이 DID 문서가 정확하게 표현되도록 하기 위한 기본 표현 기술을 소개한다. DID 문서의 주요 표현은 RFC8259에 정의된 JSON 규칙을 따르거나 CBOR 규칙을 따를 것을 추천한다. JSON 포맷을 활용 시 데이터의 상호 연동을 위해서 JSON LD를 기본적으로 사용하고 반드시 @context 속성을 포함하도록 한다.

CBOR(Concise Binary Object Representation)은 RFC7049에 정의된 구조화된 데이터가 연동가능하도록 규칙 세트를 정의한 것인데 간결하고 기계도 읽을 수 있는 언어 독립적인 데이터 교환 형식이다. 상호 연동을 위해서도 내부에 의미를 표현할 수 있도록 하며 JSON 유형도 지원한다.

5) 메소드(Methods)

DID 메소드는 각기 다른 검증가능한 저장소에 DID를 구현하도록 하기 위한 방법을 의미한다. 새로운 DID 메소드들은 해당 시스템의 스펙 내에 정의되어 같은 메소드를 사용하는 서로 다른 구현체에서 상호 연동이 가능하도록 해야 한다.

본 섹션에서는 이러한 메소드들의 스키마와 메소드가 제공하는 기본 기능, DID와 DID 문서의 생성, 조회, 업데이트, 비활성화에 대한 부분을 어떻게 정의해야 하는지 설명한다.

6) 분해(resolution)

분해(resolution)는 DID URL을 통해 DID 문서를 조회할 수 있도록 하는 기능을 의미한다. 이것은 메소드의 조회(read) 기능을 통해 가능하다. 본 섹션에서는 분해에 대한 기본적인 사항을 설명하고 분해에 필요한 메타 데이터 속성, 메타 데이터 구조에 대해 설명한다.

그러나 W3C DID 표준에서 분해에 대한 부분은 해당 scope 자체는 아니다.

5. 탈중앙 ID v.1.0 핵심 아키텍처, 데이터 모델 및 표현 해설서

- 이 문서의 버전:
  https://www.w3.org/TR/2020/WD-did-core-20201108/

- 발행된 최신 버전:
  https://www.w3.org/TR/did-core/

- 편집자 최신 원고(draft):
  https://w3c.github.io/did-core/

- 이전 버전:
  https://www.w3.org/TR/2020/WD-did-core-20201104/

- 편집자:
  Drummond Reed(Evernym)
  Manu Sporny(Digital Bazaar)
  Markus Sabadello(Danube Tech)

- 이전 편집자:
  Drummond Reed(Evernym)
  Manu Sporny(Digital Bazaar)
  Dave Longley(Digital Bazaar)
  Christopher Allen(Blockchain Commons)
  Ryan GrantMarkus Sabadello(Danube Tech)
  Jonathan Holt, DO, MS(ConsenSys Health)

ㅇ 표준 참여하기:

GitHub w3c/did-core

File a bug

Commit history

Pull requests

요약

탈중앙 identifiers(이하 DIDs)는 검증 가능하고 분산된 디지털 인증을 가능하게 하는 새로운 유형의 인증 ID입니다. DID는 DID 컨트롤러를 통해 어떤 주체(예, 개인, 기업, 사물, 데이터 모델, 추상적인 엔터티 등)를 식별합니다. 일반적인 인증 ID와 달리, DIDs는 중앙 등록 기관, ID 공급기관, 인증기관으로 분리되어 서비스 되도록 설계되었습니다. 특별히 DID 컨트롤러는 각각의 기관들이 DID와 관련된 내용을 식별하는 데 도움을 줄 수 있지만, 다른 기관의 허가 없이도 DID를 제어할 수 있습니다. DIDs는 URL로 구성되며 이 URL들은 DID 주체(DID subject)와 관련이 있습니다. DID 주체(DID subject)는 타 주체들과 신뢰할 만한 상호작용을 할 수 있도록 하는 DID 문서를 보유하고 있습니다. 각 DID 문서는, 암호화된 자료, 인증 방법 또는 서비스 엔드 포인트를 표현할 수 있으며, DID 컨트롤러가 DID를 증명하도록 하는 메커니즘을 제공합니다. 서비스 엔드 포인트는 해당 DID 주체와 신뢰할 수 있는 상호작용을 할 수 있게 합니다. DID 문서는 해당 DID 주체 자제(예, 데이터 모델)에 대한 시맨틱 기반의 의미를 포함할 수도 있습니다. 또 DID 문서는 DID 주체 그 자체(예를 들어 데이터 모델)를 포함할 수도 있습니다.

이 문서는 DIDs와 DID 문서, DID 메소드들과 URL 포맷, 공통 데이터 모델을 정의합니다.

이 문서의 상태(Status of This Document)

이 섹션에서는 발행 당시의 이 문서 상태에 대해 설명합니다. 다른 문서가 이 문서를 대체 할 수도 있습니다. 현재 W3C 간행물 목록과 이 기술 보고서의 최신 개정판은 https://www.w3.org/TR/의 W3C 기술 보고서 목록에서 찾을 수도 있습니다.

본 표준은 현재 개발 중이며 W3C DID 작업 그룹에 직접 관여하지 않는 한 표준을 실제 서비스에는 구현하지 않는 것이 좋습니다. 이 문서에 대한 요구 사항은 [DID-USE-CASES]에 정의되어 있습니다.

현재 40개 정도의 실험적인 구현체가 있으며, 최종적으로 본 표준을 구현할 수 있는지 적합성 여부를 판단하기 위한 예비 테스트 스위트(suite)가 있습니다. 본 문서를 읽는 분들은 Appendix A에 있는 주요한 이슈에 대해 주의를 기울이기를 바랍니다. 해당 본문에 우려되는 사항과 변경되는 사항들이 기록되어 있습니다.

이 문서에 대한 의견을 환영합니다. GitHub에 직접 문제를 제출하거나 public-did-wg@w3.org (subscribe, archives)로 제출하기 바랍니다.

이 사양에 대한 작업의 일부는 HSHQDC-16-R00012-H-SB2016-1-002 및 HSHQDC-17-C-00019 계약에 따라 미국 국토안보부 과학기술국의 자금 지원을 받았습니다. 이 사양의

내용은 반드시 미국 정부의 입장이나 정책을 반영하는 것은 아니며 공식적인 미국 정부의 서명이 있는 것은 아닙니다.

이 문서에 대한 작업은 Christopher Allen, Shannon Appelcline, Kiara Robles, Brian Weller, Betty Dhamers, Kaliya Young, Kim Hamilton Duffy, Manu Sporny, Drummond Reed, Joe Andrieu,, Heather Vescent가 추진한 Rebooting the Web of Trust 커뮤니티에서도 도움을 주었습니다.

이 사양에 대한 논의는 GitHub Issue를 통해 활용해 주시거나 메일 리스트에 의견을 보낼 수 있습니다. public-did-wg@w3.org(archives)

작업 초안으로 게시한다고 해서 W3C 승인을 의미하지는 않습니다. 본 문서는 초안 문서이며 언제든지 다른 문서에 의해 업데이트, 대체 또는 폐기 될 수 있습니다. 따라서 이 문서를 완성된 문서에 인용하는 것은 부적절합니다.

이 문서는 W3C 특허 정책에 따라 운영되는 워킹 그룹에서 작성했습니다. W3C는 해당 워킹 그룹의 결과물과 관련하여 만들어진 모든 특허 공개 목록을 유지합니다. 이 페이지에는 특허 공개 지침도 포함되어 있습니다. W3C 특허 정책 섹션 6에 따라, 주요 청구항을 포함하는 특허에 대한 실제 지적 재산을 가진 개인은 해당 정보를 공개해야 합니다.

이 문서는 2019년 3월 1일 W3C 프로세스 문서의 적용을 받습니다.

1) 소개

이 섹션은 비규범적입니다.

우리 중 많은 사람들은 개인 및 조직으로서 다양한 컨텍스트에서 고유한 글로벌 식별자를 사용합니다. 통신주소(전화 번호, 이메일 주소, 소셜 미디어의 사용자 이름), ID 번호(여권, 운전면허증, 세금 ID, 건강 보험 용) 및 제품 식별자(일련번호, 바코드, RFID)가 그것들입니다. 인터넷의 리소스는 MAC 주소 형식의 고유한 글로벌 식별자로 식별됩니다. URI(Uniform Resource Identifier)는 웹의 리소스에 사용되며 브라우저에서 보는 각 웹 페이지에는 전역적으로 고유한 URL(Uniform Resource Locator)이 있습니다.

이러한 글로벌 고유 식별자의 대부분은 우리가 통제하지 않습니다. 누가 또는 무엇을 식별하고 언제 취소할 수 있는지를 결정하는 외부 기관에서 이것들을 통제하고 ID를 발행합니다. 발행된 식별자들은 특정 콘텍스에서만 유용하며(우리가 선택한 것이 아닌) 특정 기관에서만 인식됩니다. 기업의 파산에 따라 이러한 식별자들은 사라지거나 유효하지 않게 될 수 있습니다. 불가피하게 개인정보가 모두 공개될 수도 있습니다. 악의적인 제 3자에 의해 복제되고 사용되는 경우도 상당히 많습니다("신원 도용").

이 사양에 정의된 탈중앙 ID(DID)는 개인이나 조직이 우리가 신뢰하는 시스템을 사용하여 자체 식별자를 생성하고(디지털 서명, 개인정보 보호 생체 인식 프로토콜 등과 같은) 암호화 증명을 사용하여 해당 식별자(인증)를 조절할 수 있도록 설계된 새로운 유형의 글로벌 고유 식별자입니다.

다른 서비스 제공자가 아닌 사용자가 직접 이러한 식별자를 생성하고 증명하는 것을 제어하기 때문에, 사용자 각자는(이 단어의 일상적인 의미에서) ID, 페르소나 및 컨텍스트를 분리되도록 설계하는데 필요한 만큼 많은 DID를 가질 수 있습니다. 이러한 식별자의 사용 범위를 가장 적절한 컨텍스트로 지정할 수 있습니다. 우리는 지속적인 식별자를 증명하는 중앙조직에 의존하지 않고, 개인 또는 사적인 정보 공개에 대한 통제를 유지하면서 자신(또는 우리가 통제하는 사물)을 식별하는데 필요한 다른 사람, 기관 또는 시스템과 상호작용할 수 있습니다.

본 문서는 DID의 생성, 지속성, 해결 또는 해석을 뒷받침하는 특정 기술이나 암호화기술 등을 요구하지 않습니다. 대신 다음을 정의합니다.

a) 모든 DID에 대한 일반 구문 및

b) (DID 문서라고 불리는)DID와 관련된 메타 데이터에 대해 네 가지 기본 CRUD 작업(생성, 읽기, 업데이트, 삭제)을 수행하기 위한 일반 요구 사항.

이를 통해 개발자는 신뢰할 수 있는 컴퓨팅 인프라(예 : 블록 체인, 분산 원장, 분산 파일 시스템, 분산 데이터베이스, P2P 네트워크)와 함께 작동하도록 특정 유형의 DID를 설계할 수 있습니다.

특정 유형의 DID에 대한 사양을 DID 메소드이라고 합니다. DID를 사용하는 애플리케이션 또는 시스템의 개발자는 특정 사용 사례에 가장 적합한 DID 방법을 지원하도록 선택할 수 있습니다.

본 문서는 다음의 사용자를 위한 것입니다.

- 시스템 사용자가 자신의 고유 식별자를 생성하고 주장할 수 있도록 하려는 개발자(DID 생산자)
- 시스템에서 사용자가 제어하는 식별자를 허용하려는 개발자(DID 소비자)
- 특정 컴퓨팅 인프라에서 DID를 사용하려는 개발자(DID 메소드 개발자).

참고 : DID 시스템의 다양성

통합 또는 중앙 집중식 ID 관리 시스템에 등록된 식별자에 대해서도 DID 메소드를 개발할 수도 있습니다. 실제로 거의 모든 유형의 식별자 시스템이 DID를 지원할 수도 있습니다. 따라서 중앙 집중식, 연합식 및 분산식 식별자의 세계 사이에 상호운용도 가능합니다.

(1) 간단한 예

이 섹션은 비규범적입니다.

DID는 다음 세 부분으로 구성된 간단한 텍스트 문자열입니다.

- URI 체계 식별자(did)
- DID 메소드를 위한 식별자
- DID 특정 메소드 식별자

Example 1 : 탈중앙 ID(DID)의 간단한 예

```
did : example : 123456789abcdefghi
```

위의 DID 예제는 DID 문서를 통해 분해됩니다. DID 문서는 DID 주체와 연동하기 위해 사용되는 서비스뿐만 아니라 암호로 DID 컨트롤러를 인증하기 위한 방법과 같은 DID 관련 정보를 포함하고 있습니다.

Example 2 : 최소한의 DID 문서 샘플

```
{
  "@context": "https://www.w3.org/ns/did/v1",
  "id": "did : example : 123456789abcdefghi",
  "authentication": [{
    //used to autjenticate as did:…fghi
    "id": "did : example : 123456789abcdefghi # keys-1",
    "type": "Ed25519VerificationKey2018",
    "controller": "did : example : 123456789abcdefghi",
    "publicKeyBase58":
"H3C2AVvLMv6gmMNam3uVAjZpfkcJCwDwnZn6z3wXmqPV"
  }],
  "service": [{
    //used to review Verifiable Credentials associated with the DID
    "id": "did : example : 123456789abcdefghi # vcs",
    "type": "VerifiableCredentialService",
    "serviceEndpoint": "https://example.com/vc/"
  }]
}
```

(2) 디자인 목표

이 섹션은 비규범적입니다

DID는 '검증 가능한 자격 증명 에코시스템 [VC-DATA-MODEL]'과 같은 대규모 시스템의 구성 요소이며 이 문서는 그 시스템을 설계하는 것이 목표입니다. 이 기본 설계 목표는 다음과 같습니다.

목표	설명
분산 (decentralization)	글로벌 단일 식별자, 공개 키, 서비스 엔드 포인트, 그 밖의 메타 데이터를 포함하여 집중화된 관리에 대한 요구 사항은 배제하는 것
제어 (control)	인간 및 비인간 엔티티에게 외부 기관에 의존할 필요없이 디지털 식별자를 직접 제어할 수 있는 권한을 부여하도록 하는 것
개인정보보안 (privacy)	엔티티가 속성 또는 다른 데이터의 최소, 선택적, 점진적 공개를 포함하여 정보의 개인정보를 스스로 제어할 수 있도록 하는 것
보안 (security)	요청 당사자가 필요한 수준의 보증을 위해 DID 문서에 의존하도록 충분한 보안을 제공하는 것
증명 기반 (proof-based)	다른 엔티티와 상호작용할 때 DID 컨트롤러가 암호화 증거를 제공할 수 있도록 하는 것
발견 가능성 (discoverability)	엔티티가 다른 엔티티에 대한 DID를 검색하고 해당 엔티티에 대해 자세히 알아 보거나 상호작용할 수 있도록 하는 것
상호 운용성 (interoperability)	상호 운용 가능한 표준을 사용하여 DID 인프라가 연동을 위해 설계된 기존 도구 및 소프트웨어 라이브러리를 사용할 수 있도록 하는 것
휴대성 (portability)	시스템 및 네트워크에 독립적이어야 하며 엔티티가 DID 및 DID 메소드를 지원하는 모든 시스템에서 디지털 식별자를 사용할 수 있도록 하는 것
단순성 (simplicity)	기술을 보다 쉽게 이해, 구현 및 배포할 수 있도록 축소된 간단한 기능 세트 제공하는 것
확장성 (extensibility)	가능한 경우 상호 운용성, 이식성 또는 단순성을 크게 저해하지 않는 경우 확장 가능하도록 하는 것

(3) 아키텍처 개요

이 섹션에서는 DID 아키텍처의 주요 요소에 대한 기본적인 이해를 제공합니다. 용어의 공식적인 정의는 [2. 용어] 섹션에서 확인 가능합니다.

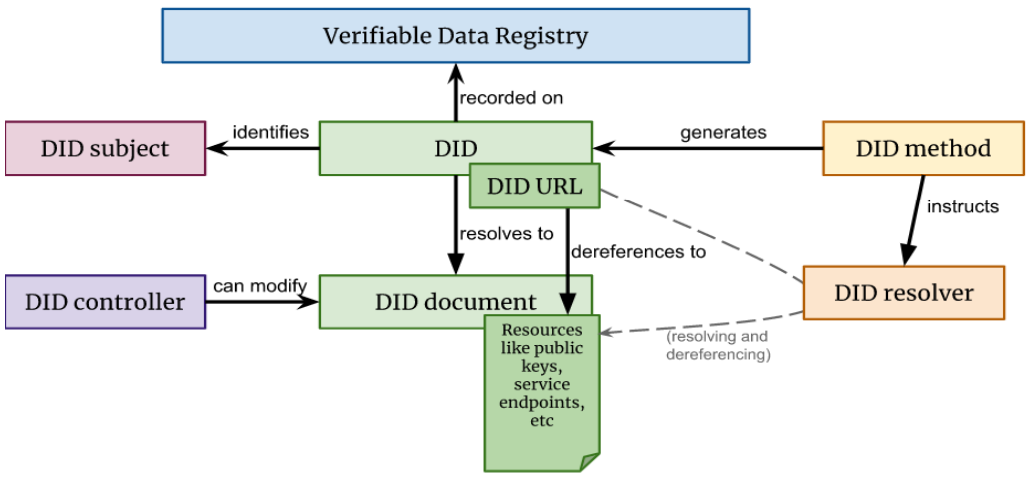

그림1. DID 아키텍처의 기본 구성

DID 및 DID URLs

DID 혹은 탈중앙 ID는 “did” 스키마, 메소드 식별자(method identifier) 및 ‘DID method’에 의해 생성된 고유의 ‘method 특정 식별자’ 등 세 부분으로 구성된 URI입니다. DID는 DID 문서들로 분해(resolution)될 수 있습니다. DID URL은 DID 문서들에 외부적으로 사용 가능한 리소소나 DID 문서 내의 공개 키와 같은 특정 리소스의 위치를 특정하기 위해 기본 DID 구문을 다른 표준 URL 컴포넌트(경로, 쿼리, 프레그먼트)에 결합하도록 확장될 수 있습니다.

DID 주체

DID 주체는 DID로 식별되는 요소라고 정의할 수 있습니다. DID 주체는 또한 DID 컨트롤러가 될 수도 있습니다. 개인, 그룹, 조직, 물리적 사물, 논리적 사물 등 무엇이든 DID의 주체가 될 수 있습니다.

DID 컨트롤러

DID 컨트롤러는 DID method에 정의된 것과 같이 DID 문서를 변화시킬 수 있는 권한을 갖춘 (개인, 조직 및 독립된 소프트웨어) 같은 요소입니다. 이 권한은 일반적으로 컨트롤러를 대신하여 작동하는 소프트웨어에서 사용하는 암호화 키 세트의 제어에 의해 확인되지만 다른 메커니즘을

통해 확인될 수도 있습니다. DID는 둘 이상의 컨트롤러를 가질 수 있으며 컨트롤러는 DID 객체를 포함할 수도 있습니다.

검증 가능한 데이터 저장소(Registries)

DID가 DID 문서로 분해될 수 있도록 하기 위하여, DID는 일반적으로 어떤 종류의 기본 시스템 또는 네트워크에 기록됩니다. 사용된 특정 기술에 관계없이 DID 기록 및 DID 문서 생성에 필요한 데이터 반환을 지원하는 모든 시스템을 검증 가능한 데이터 저장소라고 합니다. 예를 들면, 분산 원장, 분산 파일 시스템, 모든 종류의 데이터베이스, P2P 네트워크 및 기타 형태의 신뢰할 수 있는 데이터 저장소가 그것입니다.

DID 문서

DID 문서는 DID와 관련된 메타 데이터를 포함합니다. 일반적으로 DID 객체와의 상호작용과 관련된 검증 방법(예: 공개 키) 및 서비스를 표현합니다. DID 문서는 특정 구문에 따라 직렬화됩니다(§ 6. 핵심 표현을 참고하세요.). DID 자체는 id 속성값입니다. DID 문서에서 지원되는 일반 속성은 § 5. 핵심속성에 정의되어 있습니다. DID 문서에 있는 속성은 § 7. 방법에 설명된 적용 가능한 작업에 따라 업데이트될 수 있습니다.

DID 메소드

DID 메소드는 특정 유형의 DID 및 관련 DID 문서를 특정 검증 가능한 데이터 저장소를 사용하여 생성, 확인, 업데이트 및 비활성화하는 메커니즘 입니다. DID 메소드는 별도의 DID 메소드 사양에 정의됩니다(§ 7. 메소드).

주의

개념적으로 이 사양과 DID 메소드 사양 간의 관계는 IETF 일반 URI 사양([RFC3986])과 특정 URI 스키마([IANA-URI-SCHEMES](예 : http : 및 https : [RFC7230]에 지정된 스키마) 간의 관계와 유사합니다. IETF 일반 URN 사양([RFC8141])과 특정 URN 네임 스페이스 정의(예 : [RFC4122에 정의된 UUID URN 네임 스페이스) 간의 관계와 유사합니다. 차이점은 특정 DID 체계를 정의할 뿐만 아니라 DID 방법 사양이 특정 유형의 검증 가능한 데이터 저장소를 사용하여 DID 및 DID 문서를 생성, 해결, 업데이트 및 비활성화하는 방법도 지정한다는 것입니다.

DID 분해자(resolvers) 및 DID 분해(resolution)

DID 분해자(resolver)는 입력으로 DID(및 연관된 메타 데이터 입력)를 받고 출력으로 DID 문서(및 관련 메타 데이터)를 생성하는 소프트웨어 및/또는 하드웨어 구성 요소입니다. 이 프로세스를 DID 분해(resolution)라고 합니다. DID 분해 프로세스의 입력 및 출력은 § 8. 분해 방법에 정의되어 있습니다. 특정 유형의 DID를 분해하기 위한 특정 단계는 관련 DID 메소드 사양에 의해 정의됩니다. DID 분해자(resolver) 구현에 대한 추가 고려 사항은 [DID-RESOLUTION]에서 설명합니다.

DID URL 역참조자(dereferncecer) 및 DID URL 역참조

DID URL 역참조자는 입력으로 DID URL(및 연관된 메타 데이터 입력)을 받고 출력으로 리소스(및 관련 메타 데이터)를 생성하는 소프트웨어 및/또는 하드웨어 구성 요소입니다. 이 프로세스를 DID URL 역참조라고 합니다. DID URL 역참조 프로세스의 입력 및 출력은 § 8.2 DID URL 역참조에 정의되어 있습니다. DID URL 역참조자를 구현하기 위한 추가 고려 사항은 [DID-RESOLUTION]에서 설명합니다.

(4) 적합성

비규범(non-normative)으로 표시된 섹션뿐만 아니라 이 사양의 모든 작성 지침, 다이어그램, 예제 및 참고 사항은 비규범적입니다. 이 밖의 다른 모든 내용은 규범적입니다.

이 문서에서 키워드 MAY, MUST, MUST NOT, OPTIONAL, RECOMMENDED, REQUIRED, SHOULD, SHOULD NOT은 BCP 14[RFC2119] [RFC8174]에 설명된 대로 해석되어야 합니다.

이 문서에는 JSON, CBOR 및 JSON-LD 콘텐츠가 포함된 예제가 포함되어 있습니다. 이러한 예제 중 일부에는 인라인 주석(//) 및 예제에 약간의 값를 추가하는 정보를 나타내는 줄임표(…) 같이 무효한 문자가 포함되어 있습니다. 개발자는 정보를 유효한 JSON, CBOR 또는 JSON-LD로 사용하려는 경우 이 내용은 제거해야 합니다.

DID 및 DID 문서에 대한 구현의 상호 연동성은 사양을 준수하는 DID 및 DID 문서를 생성하고 구문 분석하는 구현의 능력을 평가하여 테스트됩니다. DID 및 DID 문서의 생산자와 소비자를 위한 상호 연동성은 DID 및 DID 문서의 규격에 부합하는지를 보장함으로써 제공됩니다. DID 메소드(method)의 상호 연동성 사양은 각 DID 메소드(method) 사양의 세부 사항에 의해 제공됩니다. 웹 브라우저가 모든 알려진 URI 체계를 구현할 필요가 없는 것과 동일한 방식으로 DID와 함께 작동하는 준수 소프트웨어는 모든 알려진 DID 메소드(method)를 구현할 필요는 없습니다(그러나 주어진 DID 메소드의 모든 구현은 해당 메소드에 대해 상호 운용 가능합니다.).

호환 DID은 § 3.식별자 절에 제시된 규칙으로 구체적으로 표현하며, 반드시 해당 섹션의 관련 규범을 준수해야 합니다.

호환 DID 문서는 이 사양에 제시된 데이터 모델로 구체적으로 표현하며, 반드시 § 4. 데이터 모델 및 § 5. 핵심 속성 절에 기재된 관련 기술을 준수해야 합니다. 호환 문서에 대한 직렬화 포맷은 반드시 § 6. 핵심 표현에서 설명한 바와 같이, 결정성, 양방향으로 손실이 없어야 한다. 순응 문서는 그러한 직렬화 포맷으로 저장되거나 전송할 수도(MAY) 있습니다.

호환 DID 메소드(method)는 섹션§ 7. 방법에 제시된 관련 표준 기술을 이행합니다.

호환 생산자는 소프트웨어 및/또는 하드웨어로 실현된 알고리즘이며, 생산자가 호환 DID 혹은 호환 DID 문서를 생성하는 경우 이 사양을 잘 따릅니다. 호환 생산자는 반드시 비호환 DID 및 비호환 DID 문서를 생성해서는 안 됩니다.

호환 소비자는 호환 DID 및 호환 DID 문서를 소비하는 경우 이 사양에 따른 소프트웨어 및/또는 하드웨어로 실현된 알고리즘입니다. 호환 소비자가 비표준 DID 또는 DID 문서를 사용할 때 반드시 오류가 반환되도록 해야 합니다.

2) 용어

이 섹션은 비규범적입니다.

이 섹션에서는 이 사양과 탈중앙 ID 인프라 전체에서 사용되는 용어를 정의합니다. 해당 용어에 링크가 연결되어 있습니다.

인증(authenticate)

인증은 엔터티가 증명할 수 있는 하나 이상의 증명 방법을 사용하여 특정 비밀을 특정 속성으로 갖거나 제어할 수 있도록 하는 프로세스(일반적으로 일부 유형의 프로토콜)입니다. DID와 함께, DID 문서에 발행된 공개 키와 결합된 개인 키를 제어할 수 있도록 하는 것이 일반적인 예입니다.

탈중앙 ID(DID)

암호화 방식으로 생성 및/또는 등록되기 때문에 중앙 집중식 등록 기관이 필요하지 않은 글로벌 고유 영구 식별자입니다. DID의 일반적인 형식은 DID Core 사양에 정의되어 있습니다. 특정 DID 체계는 DID 메소드 사양 섹션에 정의되어 있습니다. 전부는 아니지만 많은 DID 메소드는 분산원장기술(DLT) 또는 다른 형태의 분산 네트워크를 사용합니다.

분산 ID 관리

탈중앙 ID를 기본적으로 사용하는 ID 관리를 의미합니다. 분산 ID 관리는 X.500 디렉토리 서비스, 도메인 이름 시스템 및 대부분의 국가 ID 시스템과 같은 전통적인 신뢰 루트를 넘어서 식별자 생성, 등록 및 할당에 대한 부분까지 가능합니다.

DID 컨트롤러

DID 문서를 변경할 수 있는 기능이 있는 엔티티입니다. DID는 하나 이상의 컨트롤러를 가질 수 있습니다. DID 컨트롤러(들)는 최상위 DID 문서에서 선택적 controller 속성을 나타낼 수 있습니다. DID 컨트롤러가 DID 주체가 될 수도 있습니다.

DID 대리자(delegate)

DID 컨트롤러가 DID 문서를 통해 DID와 관련된 확인 방법을 사용할 수 있는 권한을 부여한 엔티티입니다. 예를 들어 자녀의 DID 문서를 제어하는 부모는 자녀가 인증을 위해 자신의 개인기기를 사용하도록 허용할 수 있습니다. 이 경우 자녀는 DID 대리자입니다. 자녀의 개인 장치에는 자녀가 DID를 사용하여 인증할 수 있는 개인 암호화 자료가 포함됩니다. 그러나 자녀는 부모의 허가 없이 다른 개인 장치를 추가할 수 없습니다.

DID 문서

DID 객체 또는 DID 대리인이 자신을 인증하고 DID와의 연관하여 증명하는 데 사용할 수 있는 공개 키 및 가명 생체 인식과 같은 메커니즘을 포함하여 DID 객체를 설명하는 데이터 세트입니다. DID 문서에는 DID 객체를 설명하는 다른 속성이나 클레임도 포함될 수 있습니다. DID 문서는 § 6. 핵심 표현 또는 W3C DID 사양 레지스트리 [DID-SPEC-REGISTRIES]에 정의된 대로 하나 이상의 다른 표현을 가질 수 있습니다.

DID 프리그먼트(fragment)

첫 번째 해시 기호 문자(#) 뒤에 오는 DID URL 부분입니다. DID 프리그먼트 구문은 URI 프리그먼트 구문과 동일합니다.

DID 메소드(method)

특정 DID 체계가 특정 검증 가능한 데이터 레지스트리와 함께 작동하기 위해 구현되어야 하는 방법에 대한 정의입니다. DID 메소드는 DID 메소드 사양에 의해 정의되며, DID가 생성, 분해 및 비활성화되고 DID 문서가 작성 및 업데이트되는 정확한 작업을 지정해야 합니다. § 7. 메소드를 참조하세요.

DID 경로(path)

첫 번째 포워드 슬래시(/)를 포함하거나 시작하고 물음표(?)나 프리그먼트 해시 사인(#)(혹은 DID URL 끝)으로 마치는 DID URL 부분입니다. DID 경로 구문은 URI 경로 구문과 동일합니다. § 3.2.2 경로를 참고하세요.

DID 쿼리(query)

첫 번째 물음표 문자(?)를 포함하고 뒤따르는 DID URL 부분입니다. DID 쿼리 구문은 URI 쿼리 구문과 동일합니다. § 3.2.3 쿼리를 참조하세요.

DID 분해(resolution)

이 기능은 DID 및 입력 메타 데이터 세트를 입력하고 추가 메타 데이터를 더하여 호환 표현 방식에 따라 DID 문서를 호출하는 방식으로 이루어집니다. 이 기능은 해당 DID 메소드의 "읽기" 작업에 의존합니다. 이 함수의 입력과 출력은 § 8.분해(resolution)에 정의되어 있습니다.

DID 분해자(resolver)

입력으로 DID를 받고 출력으로 순응 DID 문서를 생성함으로 DID 분해(resolution)작용을 하는 소프트웨어 및/또는 하드웨어 구성요소입니다.

DID 스키마(schema)

탈중앙 ID의 형식적 구문입니다. 일반 DID 스키마는 DID 핵심 사양 파트에 정의된 것과 같이 접두사 did:로 시작합니다. 각 DID 메소드 사양은 특정 DID 방법과 함께 작동하는 특정 DID 체계를 정의해야 합니다. 특정 DID 메소드 스키마에서 DID 메소드 이름은 첫 번째 콜론 다음에 두 번째 콜론으로 끝나야 합니다. (예, did : example :)

DID 주체

DID로 식별되고 DID 문서로 설명되는 엔티티입니다. DID는 정확히 하나의 DID 주체를 가집니다. 개인, 단체, 조직, 물리적 사물, 디지털 사물, 논리적 사물 등 무엇이든 DID 주체가 될 수 있습니다.

DID URL

DID § 3.2 DID URL 구문 정의에 부합하는 추가 구문 요소를 추가할 수 있습니다. 여기에는 선택적 DID 경로, 선택적 DID 쿼리(및 선행 ? 문자) 및 선택적 DID 프리그먼트(및 선행 # 문자)이 포함됩니다.

DID URL 역참조(dereferencing)

입력으로 DID URL, DID 문서 및 일련의 역참조 옵션을 취하고 리소스를 반환하는 함수입니다. 이 리소스는 추가 메타 데이터가 더해진 DID 문서가 될 수도 있고 DID 문서에 포함된 보조 리소스일 수도 있고 DID 문서의 전체 외부 자원일 수도 있습니다. 함수가 DID URL로 시작하는 경우 DID 분해 기능을 사용하여 DID URL에 포함된 DID로 표시된 DID 문서를 가져옵니다. 그러면 DID URL로 표시된 역참조된 리소스를 반환하도록 DID 문서에 추가적으로 기능하게 하는 역참조 기능입니다. 이 함수의 입력과 출력은 § 8.2 DID URL 역참조에 정의되어 있습니다.

분산 원장(DLT)

다양한 노드들이 각 트랜잭션이 암호화 서명하고 이전 거래에 체인되는 공유 원장을 유지하도록 하는 합의 프로토콜을 사용하는 분산 데이터베이스입니다.

공개 키 설명(description)

공개 키 또는 확인키를 사용하는 데 필요한 모든 메타 데이터를 포함하는 DID 문서 내에 포함된 데이터 개체입니다.

리소스(resource)

[RFC3986]에 의해 HTTP 정의된 대로 : "... '리소스'이라는 용어는 URI로 식별될 수 있는 모든 것에 대해 일반적인 의미로 사용됩니다." 마찬가지로, 모든 리소스는 DID로 식별된 DID 주체로 역할할 수 있습니다.

표현(representation)

[RFC7231]에 의해 HTTP 정의된 대로 : "프로토콜을 통해 쉽게 전달할 수 있는 형식으로 주어진 리소스의 과거, 현재 또는 원하는 상태를 반영하고 표현 메타 데이터 세트로 구성된 정보 그리고 잠재적으로 제한되지 않은 표현 데이터 스트림."입니다. DID 문서는 DID 주체가 나타내는 정보의 표현(representation)입니다. DID 핵심 사양과 § 6. 핵심 표현(representation) 부분에서 DID 문서에 대한 여러 표현 형식을 정의합니다.

서비스

하나 이상의 서비스 엔드 포인트를 통해 DID 객체 또는 관련 엔티티와 통신하거나 상호작용하는 수단입니다. 예를 들면 검색 서비스, 에이전트 서비스, 소셜 네트워킹 서비스, 파일 스토리지 서비스 및 검증 가능한 자격 증명 저장소 서비스가 있습니다.

서비스 엔드 포인트

서비스가 DID 객체를 대신하여 작동하는 네트워크 주소입니다(예 : HTTP URL).

URI(Uniform Resource Identifier)

[RFC3986]에서 정의한 World Wide Web의 모든 리소스에 대한 표준 식별자 형식입니다. DID는 URI 체계 유형입니다.

검증 가능한 자격 증명

W3C [VC-DATA-MODEL]에 정의된 암호화로 검증 가능한 디지털 자격 증명에 대한 표준 데이터 모델 및 표현 형식입니다.

검증 가능한 데이터 저장소(registry)

탈중앙 ID 및 DID 문서의 생성, 확인, 업데이트 및/또는 비활성화를 용이하게 하는 시스템입니다. 검증 가능한 데이터 저장소는 검증 가능한 자격 증명과 같은 다른 암호화로 검증 가능한 데이터 구조에도 사용될 수 있습니다. 자세한 내용은 [VC-DATA-MODEL]을 참조하십시오.

검증 가능한 타임 스탬프

검증 가능한 타임 스탬프를 사용하면 제 3자가 데이터 개체가 특정 시점에 존재하고 그 시점 이후로 수정되거나 손상되지 않았는지 확인할 수 있습니다. 해당 시점 이후 데이터 통합이 합리적으로 수정되거나 손상되는 경우 타임 스탬프를 검증할 수 없습니다.

검증 방법

증명을 독립적으로 검증하기 위한 프로세스 또는 프로토콜과 함께 사용할 수 있는 매개 변수 집합입니다. 예를 들어, 디지털 서명에 대한 검증 방법으로 공개 키를 사용할 수 있습니다. 이러한 사용에서 서명자가 관련 개인 키를 소유하고 있는지 검증합니다.

"검증" 및 "증거"는 광범위하게 적용됩니다. 예를 들어, Diffie-Hellman 키 교환 중에 공개 키를 사용하여 암호화를 위한 공유 대칭 키를 협상할 수 있습니다. 이는 키 동의(key agreement) 프로세스의 통일성을 보장합니다. 따라서 프로세스 설명에 "검증" 또는 "증명"이라는 단어를 사용하지 않을 수 있지만 이는 또 다른 유형의 검증 방법입니다.

검증 관계

DID 주체와 검증 방법 사이의 관계 표현입니다. 검증 관계에 대한 예제는 § 5.4.1 인증에 나와 있습니다.

UUID(Universally Unique Identifier)

[RFC4122]에서 정의된 글로벌 고유 식별자 유형입니다. UUID는 중앙 집중식 등록 기관이 필요하지 않다는 점에서 DID와 유사합니다. UUID는 확인할 수 없거나 암호화로 증명할 수 없다는 점에서 DID와 다릅니다.

위의 용어 외에도 이 사양은 [INFRA] 사양의 용어를 사용하여 추상 데이터 모델을 공식적으로 정의합니다. string, ordered set 및 map과 같은 [INFRA] 용어가 사용되면 해당 사양에 직접 연결됩니다.

3) 식별자

이 섹션에서는 DID 및 DID URL의 형식적 구문을 설명합니다. "일반(generic)"이라는 단어는 여기에 정의된 구문과 특정 DID 메소드에 의해 정의된 구문을 구분하는 데 사용됩니다.

(1) DID 구문

일반(generic) DID 스키마(schema)는 [RFC3986]을 준수하는 URI 스키마 입니다.

DID 스키마 이름은 반드시 ASCII 소문자 문자열이어야 합니다.

DID 메소드의 이름은 반드시 ASCII 소문자 문자열이어야 합니다.

다음은 ALPHA 및 DIGIT를 정의하는 [RFC5234] 내의 구문을 사용한 ABNF 정의입니다. ABNF에 정의되지 않은 다른 모든 규칙 이름은 [RFC3986]에 정의되어 있습니다.

참고

```
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 / "." / "-"/ "_"
```

DID 메소드 스펙은 자신의 method-name과 자신의 method-specific-id 구문으로 정의함으로 일반(generic) DID 구문 사용은 제한해야 합니다. method-specific-id 값에 대한 대소문자

구분과 정규화에 대한 사항은 DID 메소드를 결정하는 데에 따라 정의되어야 합니다. 자세한 내용은 섹션 § 7. 메소드 섹션을 참조하십시오.

참고 : 지속성

DID 지속적이고 불변할 것으로 예상됩니다. 즉, DID는 유일한 주체에 배타적이고 영구적으로 바인딩됩니다. DID가 비활성화된 후에도 용도가 변경되지는 않습니다.
이상적으로 DID는 시간이 지나도 여러 기본 검증 가능한 데이터 레지스트리에 바인딩될 수 있는 완전히 추상적인 탈중앙 ID(예 : UUID)로서 특정 시스템과 독립적으로 지속성을 유지할 수 있습니다. 그러나 여러 '검증 가능한 데이터 저장소'에 동일한 식별자를 등록하면 다른 검증 가능한 데이터 저장소 간에 내용이 다른 경우 실제 권한이 있는 DID 문서를 식별하기가 매우 어렵습니다. 또한 이것은 개발자가 구현하는 데 복잡하게 만듭니다.
이러한 문제를 방지하려면 개발자는 분산된 특성 Rubric [DID-RUBRIC]을 참조하여 사용 사례의 요구에 가장 적합한 DID 메소드를 결정해야 합니다.

(2) DID URL 구문

DID URL은 항상 리소스의 위치를 식별할 수 있습니다. 예를 들어 DID 문서의 특정 부분을 식별하는 데 사용할 수 있습니다.

다음은 [RFC5234]의 구문을 사용한 ABNF 정의입니다. 이 정의는 § 3.1 DID 구문에 정의된 did 스키마 기반으로 작성되었습니다. path-abempty, query 및 fragment 컴포넌트는 [RFC3986]에 정의된 ABNF 규칙과 동일합니다.

참고

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

참고

이 사양은 [MATRIX-URIS]에 설명된 대로 매개 변수의 하위 구분 기호로 향후 사용할 수 있도록 세미콜론(;) 문자를 나타냅니다.

① DID 매개 변수

DID URL 구문은 매개 변수에 대하여 query 컴포넌트 기반의 간단한 포맷을 지원합니다. (§ 3.2.3 쿼리 참조)

몇몇 DID 매개 변수 이름(예를 들어 서비스)은 다른 특정 DID 메소드와는 완전히 독립적이고, 다른 DID들과 동일한 방식으로 작동해야만 합니다. 또 다른 DID 매개 변수 이름(예를 들어 버저닝)은 특정 DID 메소드에 의해 지원될 수도 있지만 그들을 지원하는 DID 메소드들에 대하여 동일하게 작동해야 합니다.

다음 표는 DID 매개 변수 이름 세트를 정의합니다.

매개 변수 이름	설명
hl	[HASHLINK]에 지정된 대로 통합 보호를 추가하기 위한 DID 문서의 리소스 해시입니다. 관련 값은 반드시 ASCII 문자열이어야 합니다. 이 매개 변수는 비표준입니다.
service	서비스 ID를 통해 DID 문서에서 서비스를 식별합니다. 관련 값은 반드시 ASCII 문자열이어야 합니다.
version-id	확인할 DID 문서의 특정 버전을 식별합니다(버전 ID는 순차적이거나 UUID 또는 method-specific일 수 있습니다). 이 매개 변수는 모든 DID 메소드에서 지원되지 않을 수 있습니다. 관련 값은 반드시 ASCII 문자열이어야 합니다.
version-time	확인할 DID 문서의 특정 버전 타임 스탬프를 식별합니다. 즉, 특정 시점에 DID에 대해 유효한 DID 문서입니다. 이 매개 변수는 모든 DID 방법에서 지원되지 않을 수 있습니다. 관련 값은 반드시 ASCII 문자열이어야 합니다.
relative-ref	서비스 매개 변수를 사용하여 DID 문서에서 선택된 서비스 엔드 포인트에서 리소스를 식별하는 RFC3986 섹션 4.2에 따른 상대 URI 참조입니다. 연관된 값은 반드시 ASCII 문자열이어야 하며 RFC3986 2.1에 지정된 특정 문자에 대한 퍼센트 인코딩을 사용해야 합니다.

DID 메소드 스펙 작성자 및 개발자들은 위에 나열되지 않은 추가 DID 매개 변수를 사용할 수 있습니다. 최대한의 상호 연동을 위하여 DID 매개 변수는 공식 W3C DID Specification Registries 메카니즘[DID-SPE-REGISTRIES]을 사용하도록 권고하며 이는 다른 의미의 같은 DID 매개 변수를 사용하면서 나타나는 혼란을 피하도록 합니다.

이러한 매개 변수 처리에 대한 추가 고려 사항은 [DID-RESOLUTION]에서 설명합니다.

다음은 service 및 version-time DID 매개 변수를 사용하는 두 가지 DID URL 예제입니다.

Example 3 : 'service' DID 매개 변수가 있는 DID URL

'service' DID 매개 변수가 있는 DID URL

Example 4 : 'version-time' DID 매개 변수가 있는 DID URL

did : foo : 21tDAKCERh95uGgKbJNHYp? version-time = 2002-10-10T17 : 00 : 00Z

DID URL에 DID 매개 변수를 추가한다는 것은 그 매개 변수가 DID 문서 또는 기타 리소스에 대한 식별자의 일부가 된다는 것을 의미합니다. 반대로 DID 분해와 DID URL 역참조 기능들이 DID URL의 일부분이 아닌 DID 분해자에 입력 메타 데이터를 전달하는 것에 따라 영향을 받을 수도 있습니다(§ 8.1.1 DID 확인 입력 메타 데이터 속성을 참조하시기 바랍니다). 이러한 입력 메타 데이터는 캐싱이나 분해 결과에 따른 인코딩을 제어할 수 있습니다. 이는 특정 매개 변수가 HTTP URL에 포함되거나 역참조 프로세스 중에 HTTP 헤더로 전달될 수 있는 HTTP와 유사합니다. 중요한 차이점은 DID URL의 일부가 아닌 입력 메타 데이터가 리소스를 확인하거나 역참조를 제어하는 데 사용될 수 있는 반면에 DID URL의 일부 DID 매개 변수는 식별하고자 하는 리소스를 지정하는 데 사용된다는 것입니다.

RDF/JSON-LD 문서 내 리소스로서 혹은 링크로 사용될 수 있는 URL 일부로 필요한 매개 변수를 사용할 경우 DID 매개 변수는 사용될 수 있습니다.

DID 분해자(resolver)에 입력 메타 데이터를 전달하도록 하는 동일 기능이나 자원 RDF/JSON-LD 문서 내 리소스나 링크로 사용하기 위한 URL을 구성하는 데 필요가 없는 경우 DID 매개 변수는 사용될 수 없습니다.

② 경로(path)

DID 경로는 일반적인 URI 경로와 동일해야 하고 [RFC3986]의 path-abempty ABNF 규칙에 반드시 부합해야 합니다.

DID 메소드 사양은 이 섹션 내 일반 규칙보다 더 제한적인 DID 경로(path)를 위해 ABNF 규칙을 명시해야 합니다.

Example 5

```
did : example : 123456 / path
```

③ 쿼리(Query)

DID 쿼리는 일반적인 URI 쿼리에서 파생되고, § 3.2 DID URL 구문에 있는 did-query ABNF 규칙에 반드시 부합해야 합니다. DID 쿼리가 있는 경우 § 3.2.1 DID 매개 변수에 설명된 대로 사용해야 합니다.

DID 메소드 사양은 이 섹션의 일반 규칙보다 더 제한적인 DID쿼리에 대한 ABNF 규칙을 명시해야 합니다.

Example 6

```
did : example : 123456? query = true
```

④ 프래그먼트(fragment)

DID 프래그먼트는 DID 문서나 외부 리소스를 메소드 독립적으로 참조하는 데 사용됩니다.

DID 프래그먼트 구문과 맥락은 일반(generic) URI 프래그먼트와 동일하고 반드시 RFD 3986의 섹션 3.5를 준수해야 합니다. DID 프래그먼트를 역참조하기 위해 DID 프래그먼트를 포함하는 완전한 형태의 DID URL은 반드시 DID URL 역참조 기능에 대하여 인풋으로 사용되어야 합니다.

자세한 내용은§ 8.2 DID URL 역참조를 참조하시기 바랍니다.

상호 연동을 극대화하기 위해 개발자는 DID 프래그먼트가 § 6. 핵심 표현에 설명되어 있는 표현과 동일한 방식으로 해석되도록 해야 합니다. 예를 들어, JSON 포인터 [RFC6901]은 DID 프래그먼트에서 사용될 수 있지만 표현 간에 동일한 방식으로 해석되지 않습니다.

Example 7 : DID 문서의 고유한 검증 메소드

```
did : example : 123456 # public-key-1
```

Example 8 : DID 문서의 서비스 엔드 포인트

```
did : example : 123456 # public-key-1
```

Example 9 : DID 문서의 외부 리소스

```
did : example : 123456 # public-key-1
```

이 섹션의 의미 체계와 호환되고 계층화된 프리그먼트 식별자에 대한 추가 의미 체계는 § B.2 application/did + ld + json의 JSON-LD 표현에 설명되어 있습니다.

⑤ 상대 DID URL

상대 DID URL은 did:〈method-name〉:〈method-specific-id〉로 시작하지 않는 DID 문서 내 URL 값 입니다. 보다 구체적으로, § 3.1 DID 구문에 정의된 ABNF로 시작하지 않는 모든 URL 값입니다. URL의 내용은 일반적으로 동일한 DID 문서 내 리소스를 참조합니다. 상대 DID URL은 상대적인 경로 구성 요소, 쿼리 매개 변수 및 프래그먼트 식별자를 포함하기도 합니다.

상대 DID URL 참조를 확인할 때, RFC3986 섹션 5 : 참조 확인에 지정된 알고리즘을 반드시 사용해야 합니다. 기준 URI의 값은 DID 객체와 연관된 해당 DID입니다. § 5.1 DID 객체를 참조 바랍니다. 스키마는 did입니다. 인증(authority)은 〈method-name〉:〈method-specific-id〉의 조합입니다. 경로, 쿼리 및 프래그먼트 값은 § 3.2.2 경로, § 3.2.3 쿼리, 및 § 3.2.4 Fragment에 정의된 것입니다.

상대 DID URL은 필요 이상으로 자세한 절대 URL을 사용하지 않고도 DID 문서 내 검증 방법과 서비스를 식별하는 데 자주 사용됩니다.

Example 10 : 상대 DID URL 의 예

```
{
  "@context": "https://www.w3.org/ns/did/v1",
  "id": "did : example : 123456789abcdefghi",
  "verificationMethod": [{
    "id": "did : example : 123456789abcdefghi # key-1",
    "type": "Ed25519VerificationKey2018",
    "controller": "did : example : 123456789abcdefghi",
    "publicKeyBase58": "H3C2AVvLMv6gmMNam3uVAjZpfkcJCwDwnZn6z3wXmqPV"
  }, ...],
  "authentication": [
     // a relativce DID URL used to reference a verification method above
    " # key-1 "
  ]
}
```

위의 예제에서 상대 DID URL 값은 did:example:123456789abcdefghi#key-1의 절대 DID URL 값으로 변환됩니다.

4) 데이터 모델

이 사양은 특정 표현에 관계없이 DID 문서에 대한 추상 데이터 모델을 정의합니다. 이 섹션에서는 데이터 모델에 대한 높은 수준의 설명, 표현을 위한 일련의 요구 사항 및 확장성에 대한 몇 가지의 요구 사항을 제공합니다.

(1) 정의

DID 문서는 속성 이름 및 속성값과 같은 이름-값 쌍으로 구성되어 있습니다. 이러한 각 속성의 정의는 § .5. 핵심 속성에 지정되어 있습니다. 특정 표현은 § 6. 핵심 표현에 정의되어 있습니다.

(2) 데이터 표현(representation)

다음은 데이터 표현에 대한 요구 사항입니다.

표현은 본 문서에 정의된 모든 속성 이름 및 관련 값에 대해 명확한 인코딩 및 디코딩을 정의해야 합니다. 즉, DID 문서 데이터 모델에서 표현된 모든 것은 해당 정의를 준수하는 모든 규격 표현으로 표현될 수 있다는 것입니다.

1. 표현은 반드시 IANA 등록된 MIME 유형과 연관되어야 합니다.
2. 표현은 § 3.2.4 프래그먼트에 정의된 프래그먼트 처리 규칙을 준수하는 MIME 유형에 대한 처리 규칙을 정의해야 합니다.
3. 핵심 표현은 § 6. 핵심 표현에 정의되어 있습니다.

(3) 확장성

데이터 모델은 두 가지 유형의 확장성을 지원합니다.

1. 최대한의 상호 연동을 위해서 확장을 위해서는 공식 W3C DID Specification Registries 메커니즘 [DID DID-SPEC-REGISTRIES]를 사용하기를 권장합니다. 새 속성이나 기타 확장을 위해 위 메커니즘을 사용하는 것이 두 개의 서로 다른 표현이 연동될 수 있도록 하는 유일한 방법입니다.
2. 표현은 분산된 확장 방법을 포함하여 다른 확장 메커니즘을 정의합니다. 이러한 확장 메커니즘은 반드시 다른 호환 표현 방식을 통해 손실 없는 변환을 지원해야 합니다.

참고

DID 핵심 레지스트리 [DID-SPEC-REGISTRIES]에 기록되지 않은 표현이나 확장을 상호 이해하도록 사용하는 데 합의된 범위 밖에서의 두 개의 구현체도 가능합니다. 그러나 이러한 구현과 더 큰 생태계간의 상호 연동은 신뢰도가 다소 떨어집니다.

5) 핵심 속성

DID는 DID 문서를 지시합니다. DID 문서는 § 4. 데이터 모델에 설명된 데이터 모델의 직렬화된 형태입니다. 다음 섹션에서는 이러한 속성이 DID 문서 내에서 필수인지 선택인지를 포함하여 DID 문서 내의 여러 속성을 정의 합니다. 이러한 속성은 DID 주체와 속성 값 간의 관계를 설명합니다.

참고로 DID 문서의 최상위에 있는 핵심 속성은 다음과 같습니다. DID 문서에서 참조되는 다른 주체에 속하는 속성도 각각의 최상위 속성과 함께 나열됩니다.

- id: § 5.1 DID 주체에 정의
- controller: § 5.2 컨트롤러에 정의
- verificationMethod: § 5.3 검증 방법에 정의. 하위 속성에 id, type, controller 포함
- authentication: § 5.4.1 인증에 정의
- assertionMethod: § 5.4.2 assertionMethod에 정의
- keyAgreement: § 5.4.3 keyAgreement에 정의
- capabilityDelegation: § 5.4.5 capabilityDelegation에 정의
- capabilityInvocation: § 5.4.4 capabilityInvocation에 정의
- service: § 5.5 서비스 엔드 포인트에 정의. 하위 속성에는 id, type 및 serviceEndpoint가 포함

(1) DID 주체

DID 주체는 id 속성으로 표시됩니다. DID 주체는 DID 문서에 관련된 엔티티입니다. 즉, DID에 의해 식별되고 DID 문서에 의해 설명되는 엔티티입니다.

DID 문서는 반드시 id 속성을 포함해야 합니다.

id

id 값은 반드시 § 3.1 DID 구문의 규칙을 이행하는 문자열이어야 합니다.

Example 11

```
{
  "id": "did : example : 21tDAKCERh95uGgKbJNHYp"
}
```

참고 : 중간 표현(intermediate representations)

DID 메소드 사양은 DID 분해자(resolver)가 DID 분해(resolution) 작업을 하는 동안 id 속성을 포함하지 않는 DID 문서의 중간 표현을 생성할 수도 있습니다. 그러나 완전히 확인된 DID 문서는 항상 유효한 id 속성을 포함해야 합니다. 확인된 DID 문서의 값은 반드시 확인된 DID와 일치해야 하며, 향후 확인 작업에 사용될 수 있는 DID 방법에 의해 표준화된 정식 DID로 채워야 합니다.

(2) 컨트롤(Control)

권한 부여는 DID 주체를 대신하여 수행되는 작업을 나타내는 메커니즘입니다. DID 컨트롤러는 각각의 DID 문서를 변경할 권한이 있습니다.

DID 문서는 DID 컨트롤러(들)를 나타내는 controller 속성을 포함할 수 있습니다.

컨트롤러(controller)

Controller 속성값은 반드시 § 3.1 DID 구문 규칙을 따르는 문자나 문자열이어야 합니다. 해당 DID 문서(들)는 특정 목적을 위한 특정 검증 방법의 사용을 명시적으로 허용하는 검증 관계를 포함해야 합니다.

controller 속성이 DID 문서에 존재하는 경우, 그 값은 하나 이상의 DID를 나타냅니다. 해당 DID에 대한 DID 문서에 포함된 모든 검증 방법은 권한이 있는 것으로 허용되어야 합니다. 따라서 이러한 확인 방법을 충족하는 증명은 DID 주체가 제공한 증명과 동등한 것으로 간주됩니다.

참고 : 권한 부여와(authorization) 및 인증(authentication)

권한 부여는 § .5.4.1 인증과 별개입니다. 특별히 이 권한 인증은 주체가 더 이상 키에 접근할 수 없거나 키가 손상되었을 때, 그리고 DID 컨트롤러의 신뢰할 만한 제3자가 공격자의 악의적인 활동을 극복해야 경우에 중요합니다. § 9. 보안 고려 사항을 참조 바랍니다.

Example 12 : 컨트롤러 속성이 있는 DID 문서

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

    "type": "VerifiableCredentialService",
    "serviceEndpoint": "https://example.com/vc/"
  }]
}
```

(3) 검증 방법

DID 문서는 DID 주체나 관련 당사자들과 상호작용하도록 인증하거나 허가하는 데 사용되는 암호 키와 같은 검증 방법을 나타낼 수 있습니다. 표현되는 정보에는 디지털 서명을 확인하는 데 사용할 수 있는 명확한 식별자와 공개 키 자료가 포함되어 있습니다. 예를 들어, 디지털 서명에 대한 확인 방법으로 공개 키를 사용할 수 있습니다. 이러한 경우에 서명자가 관련 개인 키를 소유하고 있는지 확인합니다.

검증 방법은 많은 매개 변수를 취할 수 있습니다. 임계 값 서명을 위해 3개가 필요한 5개의 암호화 키 세트가 한 예입니다. 메소드는 암호화할 필요가 없습니다. 후보 생체 인식 벡터에 대해 DID 컨트롤러를 비교하는 생체 인식 서비스를 제공하는 업체의 연락처 정보가 한 예일 수 있습니다.

상호 연동성을 극대화하기 위해, 검증 방법으로서 공개 키에 대한 지원은 제한됩니다. § 5.3.1 키 유형 및 형식을 참조 바랍니다. 다른 유형의 검증 방법의 경우 [DID-SPEC-REGISTRIES]에 검증 방법을 등록해야 합니다.

DID 문서에 verificationMethod 속성을 포함할 수 있습니다.

검증 방법(ValidationMethod)

DID 문서에 verificationMethod 속성이 포함되어 있는 경우, 속성 값은 반드시 검증 방법

순서를 따라야 합니다. 각 검증 방법은 속성들이 포함된 map에 기술되어 있습니다. 그 속성들은 반드시 id, type, controller 및 특정 검증 방법 속성들이 포함되어 있어야 하며, 추가적인 속성들도 포함될 수 있습니다.

검증 방법을 위한 id 속성값은 반드시 URL이어야 합니다. 하나 이상의 검증 방식이 존재하는 경우, verificationMethod 값은 반드시 같은 id에 여러 개의 엔티티를 포함할 수 없습니다. 하나의 id에 여러 개의 엔티티가 포함되어 있는 verificationMethod 값인 경우, DID 문서는 반드시 에러를 발생시킵니다.

검증 방법이 공개 키인 경우, id 속성값은 복합키로 구조화됩니다. 이것은 기존 키 관리 시스템 및 JWK [RFC7517]과 같은 키 형식과 통합하는 데 특히 유용합니다. 공개 키 지문 [RFC7638]으로 JWK kid 값이 권장됩니다. 자신의 공개 키를 JWK로 나타낸 검증 방법은 해당 프래그먼트 식별자로서 kid 값을 이용합니다. 복합키 식별자가 있는 공개 키의 사례에 대해서는 #example-13-various-public-keys의 첫 번째 키를 참조하십시오.

type 속성값은 반드시 정확히 하나의 검증 방법 유형이어야 합니다. 글로벌 상호 연동성을 최대화하기 위해, 검증 방법 유형은 [DID-SPEC-REGISTRIES]에 등록해야 합니다.

controller 속성값은 반드시 § 3.1 DID 구문 규칙에 부합하는 문자열이어야 합니다.

참고 : 검증 방법 컨트롤러 및 DID 컨트롤러

공개 키처럼 관계의 주체가 DID 문서인 경우 또는 관계의 주체가 검증 방법일 경우, 컨트롤러 속성의 의미는 동일합니다. 키는 자기 자신을 제어할 수 없고, 키 컨트롤러는 DID 문서에서 추론할 수 없기 때문에 키 컨트롤러의 ID를 명시적으로 표현해야 합니다. 검증 방법에 대한 controller 값이 반드시 DID 컨트롤러일 필요는 없다는 것이 차이점입니다. DID 컨트롤러는 DID 문서의 최상위 수준에 있는 controller 속성을 사용하여 표현됩니다. § 5.2 제어를 참조바랍니다..

Example 13 : 검증 방법 예

```
{
  "@context": ["https://www.w3.org/ns/did/v1", "https://w3id.org/security/v1"],
  "id": "did:example:123456789abcdefghi",
  ...
```

```
  "verificationMethod": [{
    "id": ...,
    "type": ...,
    "controller": ...,
    ...
  ]}
}
```

verificationMethod 속성뿐만 아니라 다양한 검증 관계와 관련된 속성에서 검증 방법을 포함하거나 참조할 수 있습니다(§ 5.4 검증 관계를 참조). 검증 방법을 참조하면 하나 이상의 검증 관계와 함께 사용할 수 있습니다.

DID 문서에서 검증 방법을 처리할 때 사용하는 단계는 다음과 같습니다.

1. 값은 verificationMethod 속성 또는 특정 검증 관계에 대한 속성과 관련된 데이터이어야 하며, null 값으로 초기화됩니다.
2. 값이 주체인 경우, 검증 방법 자료가 포함되어야 합니다. 결과를 값으로 설정합니다.
3. 값이 문자열인 경우, 검증 방법은 참고자료로 포함됩니다. 값은 URL이라고 가정합니다.
   - URL을 역참조하고 URL과 관련된 verificationMethod 속성을 검색합니다. 예를 들어, 역참조된 문서의 최상위 수준에서 verificationMethod 속성을 처리합니다.
   - 객체의 id속성이 값과 일치하면, 각 객체를 반복하면서 결과를 개체로 설정합니다.
4. 결과값에 id, type 및 controller 속성이 포함되지 않은 경우, 필수 공공 암호 자료와 마찬가지로 결과로 type이 결정된다면 오류가 발생합니다.

Example 14 : 검증 방법 포함 및 참조

```
{
 ...
  "authentication": [
    // this key is referenced, it may be used with more than one verification
relationship
    "did : example : 123456789abcdefghi # keys-1",
```

```
    //this key is embedded and may *only* be used for authentication
    {
      "id": "did : example : 123456789abcdefghi # keys-2",
      "type": "Ed25519VerificationKey2018",
      "controller": "did : example : 123456789abcdefghi",
      "publicKeyBase58":
"H3C2AVvLMv6gmMNam3uVAjZpfkcJCwDwnZn6z3wXmqPV"
    }
],
...
}
```


① 키 유형 및 형식(key types and formats)

공개 키를 검증 방법으로 사용할 수 있습니다.

검증 방법은 여러 검증 자료 속성을 포함해서는 안 됩니다. 예를 들어, publicKeyJwk와 publicKeyBase58을 동시에 사용하는 검증 방법에서 키 자료를 표현하는 것은 금지됩니다.

본 문서의 표준은 상호 연동 가능성을 높이기 위해 DID 문서에서 공개 키 자료를 표현하는 형식을 가능한 최소화합니다. 개발자가 구현해야 하는 형식의 개수가 적을수록 모든 형식을 지원할 가능성이 높아집니다. 이러한 접근 방식은 구현을 용이하게 하는 것과 그 동안 광범위하게 배포된 형식을 지원하도록 하는 것 사이에 균형을 맞추려는 것입니다. 지원되는 특정 유형의 키 형식이 본 문서에 나열되어 있습니다.

여기에 설명된 공개 키 유형을 사용할 때 공개 키 표현식은 공개 키 지원표에 나열된 것과 다른 키 형식을 사용해서는 안됩니다. 여기에 나열되지 않은 공개 키 유형의 경우 다른 검증 방법과 마찬가지로 유형 값과 해당 형식 속성을 [DID-SPEC-REGISTRIES]에 등록해야 합니다.

이슈 1

DID 워킹 그룹은 기본 인코딩 형식으로 Base58(Bitcoin) [BASE58], base64url [RFC7515]를 사용할지 또는 base16(hex) [RFC4648]을 사용할지 여전히 논쟁 중입니다. 아래 표의 항목은 현재 PEM 및 Base58(Bitcoin)을 사용할 것으로 가정하지만 워킹 그룹의 합의에 따라 base64url 및/또는 base16(hex)으로 변경될 수 있습니다.

이슈 2

DID 워킹 그룹은 secp256k1 Schnorr 공개 키 값을 이 사양에서 자세히 설명할 것인지, 그렇다면 표현 및 인코딩 방법은 어떻게 할 것인지에 대해 현재 논의하고 있습니다.

이 표는 DID 문서에서 공개 키 표현에 대한 지원을 정의합니다. 각 공개 키 유형에 대해 최대 2개의 인코딩 형식이 지원됩니다.

키 유형(type 값)	지원
RSA (RsaVerificationKey2018)	RSA 공개 키 값은 JWK[RFC7517]로 인코딩되거나 publicKeyPem 속성을 사용하여 개인정보 강화메일(PEM) 형식으로 인코딩되어야 합니다.
ed25519 (Ed25519VerificationKey2018)	Ed25519 공개 키 값은 JWK[RFC7517]로 인코딩되거나 publicKeyBase58 속성을 사용하여 Base58 비트코인 형식 [BASE58]의 원시 32바이트 공개 키값으로 인코딩되어야 합니다.
secp256k1-koblitz (보류 중)	Secp256k1 Koblitz 공개 키값은 JWK [RFC7517]로 인코딩되거나 publicKeyBase58속성을 사용하여 Base58 비트코인 형식[BASE58]의 원시 33 바이트 공개 키값으로 인코딩되어야 합니다.
secp256r1 (SchnorrSecp256k1Verification Key2019)	Secp256r1 공개 키 값은 JWK [RFC7517]로 인코딩되거나 publicKeyBase58속성을 사용하여 Base58 비트코인 형식 [BASE58]로 인코딩된 원시 32 바이트 공개 키 값으로 인코딩되어야 합니다.
Curve25519 (X25519KeyAgreementKey2019)	Curve25519(X25519라고도 함) 공개 키 값은 JWK[RFC7517]로 인코딩되거나 publicKeyBase58 속성을 사용하여 Base58 비트코인 형식 [BASE58]의 원시 32 바이트 공개 키 값으로 인코딩되어야 합니다.

Example 15 : 다양한 공개 키

```
{
  "@context": [ "https://www.w3.org/ns/did/v1", "https://w3id.org/security/v1"],
  "id": "did : example : 123456789abcdefghi",
  ...
  "verificationMethod": [{
    "id": "did : example : 123 #
_Qq0UL2Fq651Q0Fjd6TvnYE-faHiOpRlPVQcY_-tA4A",
    "type": "JwsVerificationKey2020",
    "controller": "did : example : 123",
    "publicKeyJwk": {
      "crv": "Ed25519",
      "x": "VCpo2LMLhn6iWku8MKvSLg2ZAoC-nlOyPVQaO3FxVeQ",
      "kty": "OKP",
      "kid": "_Qq0UL2Fq651Q0Fjd6TvnYE-faHiOpRlPVQcY_-tA4A"
    }
  }, {
    "id": "did : example : 123456789abcdefghi # keys-1",
    "type": "Ed25519VerificationKey2018",
    "controller": "did : example : pqrstuvwxyz0987654321",
    "publicKeyBase58":
"H3C2AVvLMv6gmMNam3uVAjZpfkcJCwDwnZn6z3wXmqPV"
  }, {
    "id": "did : example : 123456789abcdefghi # keys-2",
    "type": "Secp256k1VerificationKey2018",
    "controller": "did : example : 123456789abcdefghi",
    "publicKeyHex":"02b97c30de767f084ce3080168ee293053ba33b235d7116a
3263d29f1450936b71"
  }],
  ...
}
```

DID 문서에 공개 키가 존재하지 않는 경우, 반드시 키는 취소되거나 효력이 없어진다고 간주합니다. DID 문서는 반드시 검증 관계를 사용하여 키를 취소해야 합니다. 각 DID 메소드 사양에는 해지하는 방법과 추적 방법을 기술해야 합니다.

참고

DID 문서에서 키의 캐싱 및 만료는 전적으로 DID 분해자(resolver) 및 기타 클라이언트의 책임입니다. 자세한 내용은 § 8. 확인(Resolution)을 참조하십시오.

(4) 검증 관계

검증 관계는 DID 주체와 검증 방법간의 관계를 나타냅니다.

DID 문서는 특정 검증 관계를 나타내는 속성을 포함할 수 있습니다. 글로벌 상호 연동을 극대화하기 위해 속성은 [DID-SPEC-REGISTRIES]에 등록되어야 합니다.

DID 문서 내 정보는 반드시 DID 주체와 검증 방법간 검증 관계에 대해 명시합니다. 특정 검증 관계와 연관되지 않은 검증 방법은 반드시 검증 관계에 사용되어서는 안 됩니다. 자세한 검증 관계 예는 아래 섹션을 참조하십시오.

① 인증(authentication)

인증은 엔터티가 DID 주체임을 증명하거나 DID 컨트롤러로서 DID 주체를 대신해서 활동하는 것을 증명하는 데 사용하는 검증 관계입니다. 인증 시 검증자는 인증 당사자가 유효한 인증 증명을 제시하고 있는지, 즉 그 주체가 그 검증에서 실제로 검증을 요구하는 주체인지 확인할 수 있습니다. 성공적인 인증은 그 자체로 권한을 부여하거나 부여하지 않을 수도 있습니다. 그것은 검증 응용 프로그램에 달려 있습니다.

참고 : 인증의 활용

인증이 완료되면 해당 정보로 수행할 작업을 결정하는 것은 DID 메소드 또는 기타 응용 프로그램에 달려 있습니다. 예를 들어 특정 DID 메소드는 DID 문서를 업데이트하거나 삭제하는 데 DID 컨트롤러로 인증하는 것으로 충분하다고 결정할 수 있습니다. 다른 DID 메소드는 인증에 사용된 것과 다른 DID 문서를 업데이트하거나 삭제하기 위해 다른 키 또는 완전히 다른 검증 방법을 제시해야 할 수도 있습니다. 즉, 인증 확인 후 수행되는 작업은 DID 데이터

모델의 범위를 벗어나지만 DID 메소드 및 응용 프로그램은 이를 스스로 정의해야 합니다.

DID 문서는 authentication 속성을 포함합니다. authentication 속성은 DID 주체 및 공개 키와 같은 검증 방법 간의 관계입니다. DID 주체가 인증을 위해 일부 검증 방법 세트(authentication 속성값에 따라)를 승인했음을 의미합니다.

인증

관련 값은 순서가 지정된 하나 이상의 검증 방법들이어야 합니다. 각각의 검증 방법은 포함되거나 참조될 수 있어야 합니다.

위 명제는 인증을 시도하는 엔티티가 존재하는지, 유효한 인증 증거를 제시하는지 확인이 필요한 인증 검증자에게 유용합니다. 엔티티가 DID로 식별될 때, 검증자는 "인증" 목적으로 만들어진 증거를 포함하고 있는(일부 프로토콜 관련 형식) 일부 데이터를 받고, DID 문서 내 authentication 아래 나열된 검증 방법(예, 공개 키)들을 사용하여 검증할 수 있는 증거를 확인할 수 있습니다.

DID 문서의 authentication 속성에 의해 지시된 검증 방법들은 DID 주체를 인증하는 데에만 사용될 수 있습니다. DID 컨트롤러가 DID 주체가 아닌 경우 DID 컨트롤러를 인증하기 위해서 controller(§ 5.2 Control 참조) 값과 연관된 엔터티는 본인의 DID 문서와 첨부된 인증 검증 관계를 인증해야 할 필요가 있습니다.

Example 16 : 세 가지 확인 방법이 포함된 인증 속성

```
{
  "@context": "https://www.w3.org/ns/did/v1",
  "id": "did : example : 123456789abcdefghi",
  ...
  "authentication": [
    //this method can be used to authenticate as did:…fghi
    "did : example : 123456789abcdefghi # keys-1",
    //this method can be used to authenticate as did:…fghi
    "did : example : 123456789abcdefghi # biometric-1",
    // this method is *only* authorized for authentication, it may not
    // be used for any other proof purpose, so its full description is
```

```
    // embedded here rather than using only a reference
    {
      "id": "did : example : 123456789abcdefghi # keys-2",
      "type": "Ed25519VerificationKey2018",
      "controller": "did : example : 123456789abcdefghi",
      "publicKeyBase58": "H3C2AVvLMv6gmMNam3uVAjZpfkcJCwDwnZn6z3wXmqPV"
    }
  ],
   ...
}
```

② assertionMethod

assertionMethod 속성은 DID 주체를 대신하였다는 증거를 검증하는 데 검증 방법을 사용할 수 있음을 나타내는 검증 관계를 표현합니다. 해당 증거를 검증하는 자는 DID 문서의 assertion Method 속성이 검증 방법에 포함되어 있는지를 확인함으로 증거에 사용된 검증 방법이 목적에 맞게 증거와 사용되었는지 인정받았는지 확인할 수 있습니다.

참고 : assertionMethod 사용

assertionMethod가 설정되어 있는 경우, 검증자는 권리를 행사하는 증거를 제공하는 데 사용된 검증 방법이 유효한지 assertionMethod 관계 검증에 연관되어 있는지를 입증해야 합니다. 이 속성이 유용한 경우의 예는 검증자가 검증 가능한 자격 증명을 처리하는 동안입니다. 검증하는 동안 검증자는 증명을 주장하는 데 사용된 검증 방법이 해당 DID 문서의 assertionMethod 속성과 연관되어 있는지 확인하여 검증 가능한 자격 증명이 DID 주체에 의해 서명되었는지 확인합니다.

DID 문서는 assertionMethod 속성을 포함할 수 있습니다.

assertionMethod

관련 값은 반드시 하나 이상의 순서가 있는 검증 방법 세트여야 합니다. 각각의 검증 방법이 포함되어 있거나 참조되어 있어야 합니다.

Example 17 : 두 가지 검증 방법을 포함하는 assertionMethod 속성

```
{
  "@context": "https://www.w3.org/ns/did/v1",
  "id": "did : example : 123456789abcdefghi",
  ...
  "assertionMethod": [
    //this method can be used to assert statements as didi:…fhgi
    "did : example : 123456789abcdefghi # keys-1",
    // this method is *only*authorized for assertion of statements, it may not
    //be used for any other verification relationship, so its full description is
    //embedded here rather than using only a referance
    {
      "id": "did : example : 123456789abcdefghi # keys-2",
      "type": "Ed25519VerificationKey2018",
      "controller": "did : example : 123456789abcdefghi",
      "publicKeyBase58": "H3C2AVvLMv6gmMNam3uVAjZpfkcJCwDwnZn6z3wXmqPV"
    }
  ],
   ...
}
```


③ keyAgreement

keyAgreement 속성은 엔티티가 DID 주체를 대신하여 주요 계약 프로토콜에 참여하는 데 사용할 수 있는 관계검증를 표현하는 데 사용됩니다. 키 계약 프로토콜의 상대방은 keyAgreement 관계검증을 사용하여 키 합의 프로토콜에서 사용된 검증 방법이 계정에 포함된 keyAgreement 속성과 연관되어 있는지 확인하여 DID 주체를 대신하여 키 합의 프로토콜을 수행하는 당사자가 승인되었는지 여부를 확인할 수 있습니다.

DID 문서는 keyAgreement 속성을 포함할 수 있습니다.

keyAgreement

관련 값은 하나 이상의 순서가 지정된 검증 방법 세트여야 합니다. 각각의 검증 방법은 내장될 수도 참조될 수도 있습니다.

참고 : keyAgreement 사용

키 합의 교환 중에 사용된 검증 방법이 유효하고 keyAgreement 속성과 연결되어 있는지 확인하는 것은 검증자에게 달려 있습니다. 이 속성이 유용한 경우의 예는 DID 주체를 대신하여 TLS 세션을 설정하는 동안입니다. 이 경우 상대방은 프로토콜 핸드셰이크가 DID 문서 내 keyAgreement 속성과 연결되어 있는지 확인합니다.

Example 18 : 두 가지 검증 방법을 포함하는 Key aggrement 속성

```
{
  "@context": "https://www.w3.org/ns/did/v1",
  "id": "did : example : 123456789abcdefghi",
  ...
  "keyAgreement": [
    //this method can be used to perform key agreement as did:…fghi
    "did : example : 123456789abcdefghi # keys-1",
    //this method is *only* authorized for key agreement usage, it may not
    //be used for any other verification relationship, so its full description is
    //embedded here rather than using only a reference
    {
      "id": "did : example : 123 # zC9ByQ8aJs8vrNXyDhPHHNNMSHPcaSgNpjjsBYpMMjsTdS",
      "type": "X25519KeyAgreementKey2019",
      "controller": "did : example : 123",
      "publicKeyBase58": "9hFgmPVfmBZwRvFEyniQDBkz9LmV7gDEqytWyGZLmDXE"
    }
  ],
  ...
}
```

④ capabilityInvocation

capabilityInvocation 속성은 엔티티가 DID 주체 혹은 DID 주체를 대체하여 기능을 호출할 수 있도록 관계 검증을 표현하는 데 사용됩니다. DID 주체가 수행할 권한이 있는 기능을 표현합니다. 기능 호출 시 검증자는 DID 문서의 capabilityInvocation 속성이 포함된 증명에 사용된 검증 방법을 확인하여 기능의 유효성을 확인할 수 있습니다 .

DID 문서는 capabilityInvocation 속성을 포함할 수 있습니다.

capabilityInvocation

관련 값은 하나 혹은 하나 이상의 순서가 지정된 검증 방법 세트여야 합니다. 각각의 검증 방법은 내장되거나 참조될 수 있습니다.

참고 : capabilityInvocation의 사용

기능을 표시할 때 사용된 검증 방법이 호출되고 capabilityInvocation 속성과 연결되어 있는지 확인하는 것은 검증자의 책임입니다. 이 속성이 유용한 경우의 예는 DID 객체가 검증 방법과 StartCar 기능의 결합된 사용을 통해 차량 시동 기능을 호출하기로 선택한 경우입니다. 이 예에서 차량은 검증자가 되고 검증 방법에 capabilityInvocation 속성이 존재하는지 확인해야 합니다.

Example 19 : 두 가지 확인 방법을 포함하는 capabilityInvocation 속성

```
{
  "@context": "https://www.w3.org/ns/did/v1", "id":
  "did : example : 123456789abcdefghi",
  ...
  "capabilityInvocation : [
    //this method can be used to invoke capability as did:…fghi
    "did : example : 123456789abcdefghi # keys-1",
    //this method is *only* authorized for capability invocation usage, it may not
    //be used for any other verification relationship, so its full description is
    //embedded here rather than using only a reference
    {
```

```
      "id": "did : example : 123456789abcdefghi # keys-2",
      "type": "Ed25519VerificationKey2018",
      "controller": "did : example : 123456789abcdefghi",
      "publicKeyBase58": "H3C2AVvLMv6gmMNam3uVAjZpfkcJCwDwnZn6z3wXmqPV"
      }
  ],
    ...
}
```

⑤ capabilityDelegation

capabilityDelegation 속성은 엔티티가 다른 기능 호출자에게 DID 주체로 또는 DID 주체 대체로 기능을 부여하는 데 사용할 수 있는 관계 증명를 표현하는 데 사용됩니다. Capability Delegation 시도 시 검증자는 증거를 검증하는 방법에 DID 문서의 capabilityDelegation 속성이 포함되었는지 확인함으로써 기능 호출할 수 있는 기능 유효성을 확인할 수 있습니다. DID 문서는 capabilityDelegation 속성을 포함할 수 있습니다.

capabilityDelegation

관련 값은 하나 이상의 순서가 지정된 검증 방법 세트여야 합니다. 각각의 검증 방법은 내장될 수도 참조될 수도 있습니다.

참고 : capabilityDelegation 사용

기능을 제공할 때 사용된 검증 방법이 유효하고 capabilityDelegation 속성과 연결되어 있는지 확인하는 것은 검증자에게 달려 있습니다. 이 속성이 유용한 경우의 예는 DID 객체가 검증 방법과 StartCar 기능 호출자에 대한 기능의 결합된 사용을 통해 차량 시동 기능을 부여하기로 선택한 경우입니다.

Example 18 : 두 가지 확인 방법이 포함된 capability Delegation 속성

```
{
  "@context": "https://www.w3.org/ns/did/v1", "id":
```

```
  "did：example：123456789abcdefghi",
  …
  " capabilityDelegation ": [
    //this method can be used to perform capability delegation as did:…fghi
    "did：example：123456789abcdefghi # keys-1",
    //this method is *only* authorized for granting capability, it may not
    //be used for any other verification relationship, so its full description is
    //embedded here rather than using only a reference
    {
    "id": "did：example：123456789abcdefghi # keys-2",
    "type": "Ed25519VerificationKey2018",
    "controller": "did：example：123456789abcdefghi",
    "publicKeyBase58": "H3C2AVvLMv6gmMNam3uVAjZpfkcJCwDwnZn6z3wXmqPV"
    }
  ],
   …
}
```


(5) 서비스 엔드 포인트

서비스 엔드 포인트는 DID 문서에서 DID 주체 또는 관련 엔티티와 통신하는 방법을 표현하는 데 사용됩니다. DID 문서에 나열된 서비스들은 소셜 미디어 계정, 개인 웹 사이트, 이메일 주소 등의 개인정보 보존 메시징 서비스, 또는 그 이상의 공공 정보에 대한 정보를 포함할 수 있습니다. 자세한 내용은 § 10.1 개인 식별 정보(PII)를 비공개 유지를 참조하십시오. 서비스와 관련된 메타 데이터는 서비스별로 다릅니다. 예를 들어 암호화된 메시징 서비스와 관련된 메타 데이터는 메시징이 시작되기 전에 암호화된 링크를 어떻게 초기화하는지 알려줄 수 있습니다.

서비스에 대한 포인터는 service 속성을 사용하여 표현됩니다. 각 서비스는 그 자신의 보유 id 및 type 특성뿐만 아니라 서비스를 설명하는 다른 속성들 세트나 URL을 지니는 service Endpoint 속성을 지니고 있습니다.

DID 문서의 주요 목적 중 하나는 서비스 엔드 포인트를 검색하는 것입니다. 서비스 엔드 포인트는

발견, 인증, 허가 혹은 상호작용하기 위한 탈중앙 ID 관리 서비스를 포함하여 광고하고자 하는 DID 주체의 어느 유형의 서비스도 가능합니다.

DID 문서에는 service 속성을 포함할 수 있습니다.

서비스

DID 문서가 service 속성을 포함하고 있으면, 속성 값은 각 서비스 엔드 포인트가 특성 세트에 의해 기술되는 순서 없는 서비스 엔드 포인트가 될 수 있습니다. 각 서비스 엔드 포인트는 반드시 id, type 그리고 serviceEndpoint 속성을 가지고 있어야 하며, 추가 속성을 포함할 수도 있습니다.

id 속성값은 반드시 URI로 나타내야 합니다. service 값은 반드시 동일 id에 여러 엔터티를 포함해서는 안 됩니다. 이 경우, DID 문서 프로세서는 반드시 오류를 생성해야 합니다.

serviceEndpoint 속성값은 [RFC3986]에 부합하는 유효한 URL이어야 하며 [RFC3986]의 섹션 6에 기술된 규칙과 적용가능한 URL 스키마 사양의 정규화 규칙 혹은 이후 서비스 엔드 포인트로 기술될 속성들에 따라 정규화되어야 합니다.

서비스 엔드 포인트 프로토콜은 개방형 표준 사양으로 게시될 것으로 예상됩니다.

Example 21 : 다양한 서비스 엔드 포인트

```
{
  "service": [{
    "id": "did : example : 123456789abcdefghi # openid",
    "type": "OpenIdConnectVersion1.0Service",
    "serviceEndpoint": "https://openid.example.com/"
  }, {
    "id": "did : example : 123456789abcdefghi # vcr",
    "type": "CredentialRepositoryService",
    "serviceEndpoint": "https://repository.example.com/service/8377464"
  }, {
    "id": "did : example : 123456789abcdefghi # xdi",
    "type": "XdiService",
    "serviceEndpoint": "https://xdi.example.com/8377464"
  }, {
    "id": "did : example : 123456789abcdefghi # agent",
```

```
      "type": "AgentService",
      "serviceEndpoint": "https://agent.example.com/8377464"
    }, {
      "id": "did : example : 123456789abcdefghi # hub",
      "type": "IdentityHub",
      "verificationMethod": "did : example : 123456789abcdefghi # key-1",
      "serviceEndpoint": {
        "@context": "https://schema.identity.foundation/hub",
        "type": "UserHubEndpoint",
        "instances": [ "did : example : 456", "did : example : 789"]
      }
    }, {
      "id": "did : example : 123456789abcdefghi # messages",
      "type": "MessagingService",
      "serviceEndpoint": "https://example.com/messages/8377464"
    }, {
      "id": "did : example : 123456789abcdefghi # inbox",
      "type": "SocialWebInboxService",
      "serviceEndpoint": "https://social.example.com/83hfh37dj",
      "description": "my public social inbox",
      "spamCost": {
        "amount": "0.50",
        "currency": "USD"
      }
    }, {
      "id": "did : example : 123456789abcdefghi # authpush",
      "type": "DidAuthPushModeVersion1",
      "serviceEndpoint": "http://auth.example.com/did:example:123456789abcdefg"
    }]
}
```

서비스 엔드 포인트 인증과 관련된 보안 고려 사항에 대한 자세한 내용은 § 7.1 방법 체계 및 § 5.4.1 인증을 참조하십시오.

6) 핵심 표현(Core Representation)

DID 문서의 모든 구체적인 표현은 반드시 이 사양에 정의된 데이터 모델로 구문 분석할 수 있는 결정적 매핑을 사용하여 직렬화하여야 합니다. 모든 직렬화 방법은 반드시 문제 표현 안팎으로 양방향 전환하는 DID 문서 규칙들에 의해 정의되어야 합니다. 결과적으로 두 표현 간의 변환은 반드시 소스 형식을 DID 문서 모델로 구문 분석하여 수행해야 합니다(§ 4. 데이터 모델 및 § 5. 핵심 속성들에 기술되어 있습니다). 그런 다음 DID 문서 모델을 대상 표현으로 직렬화합니다. 구현 시 먼저 DID 문서 모델로 파싱하지 않고 표현(representations)들 간에 변환해서는 안 됩니다.

JSON, JSON-LD와 CBOR으로 구문 맵핑이 제공될지라도, 애플리케이션 및 서비스는 XML 또는 YAML와 같은 데이터 모델을 표현할 수 있는 임의의 다른 데이터 표현 구문을 사용할 수도 있습니다.

생산자는 반드시 문서의 메타 데이터에서 미디어 유형을 통해 사용된 문서 표현을 표시해야 합니다. 소비자는 content-type DID분해자 메타 데이터 필드를 통해 문서가 어떤 표현인지 확인해야 합니다(§ 8.1 DID분해(resolution)을 참조하시오). 소비자가 콘텐츠만으로 문서의 표현을 결정해서는 안 됩니다.

이 섹션의 생산 및 소비 규칙은 사양의 독립적인 구현과 완전히 호환되는 모든 구현에 적용됩니다. 이 스펙의 배포는 레지스트리에 없는 핸들링 속성을 위한 지역화된 규칙을 포함하여 사용자 합의된 표현들을 사용할 수 있습니다. § 4.3 확장성에서 자세한 정보를 참조하십시오.

(1) JSON

(리졸버 메타 데이터에 있는 application/did+json의 content-type에 표시된 바와 같이) 일반 JSON에 있는 DID 문서를 생산 및 소비하는 경우, 다음과 같은 규칙을 반드시 따라야합니다.

① 생산

DID 문서는 반드시 [RFC8259]에 부합되는 단일 JSON 개체여야 합니다. DID 문서의 모든 최상위 레벨 속성들은 반드시 JSON 개체 멤버의 이름으로 속성 이름을 사용하여 표현해야 합니다. 모든 확장성을 포함하여 섹션 § 4. 데이터 모델에 설명된 데이터 모델 속성값들은 다음과 같이

JSON 유형에 속성값을 매핑하여 JSON [RFC8259]로 인코딩해야 합니다.

- IEEE 754로 표현할 수 있는 숫자값은 반드시 숫자 유형(number type)으로 표현해야 합니다.
- Boolean 값은 반드시 Boolean 문자 그대로 표현되어야 합니다.
- 시퀀스값은 반드시 배열 유형(array typr)으로 표현되어야 합니다.
- 순서가 없는 값들의 세트는 반드시 배열 유형(array type)으로 표현되어야 합니다.
- 속성 세트는 반드시 객체 유형(object type)으로 표현되어야 합니다.
- 빈 값은 반드시 null 문자 그대로 표현되어야 합니다.
- 다른 값들은 반드시 문자열 유형(string type)으로 표현되어야 합니다. JSON을 생성하는 구현자는 알고리즘이 [INFRA] 사양의 JSON 직렬화 규칙과 일치하는지 확인하는 것이 좋습니다.

DID 문서의 모든 속성은 반드시 루트 개체에 포함되어야 합니다. 속성들은 위 목록의 값 표현 규칙에 따라 추가 데이터 하위 구조 객체를 정의할 수 있습니다.

@context 멤버 이름은 반드시 JSON-LD 생산자가 사용할 수 있는 속성으로 사용해서는 안 됩니다.

② **소비**

최상위 요소는 반드시 JSON 객체입니다. 최상위 레벨의 다른 데이터 유형은 오류이며 반드시 거부되어야 합니다. 최상위 JSON 객체는 DID 문서를 나타내며 이 개체의 모든 구성원은 DID 문서의 속성입니다. 개체 멤버 이름은 속성 이름이며 멤버 값은 다음과 같이 해석됩니다.

- 숫자 유형은 반드시 IEEE 754로 나타낼 수 있는 숫자 값으로 해석되어야 합니다.
- Boolean 문자 그대로 반드시 Boolean 값으로 해석되어야 합니다.
- 배열 유형은 반드시 이 값에 대한 속성의 정의에 따라 시퀀스 또는 순서 없는 세트로 해석되어야 합니다.
- 개체 유형은 반드시 속성 세트로 해석되어야 합니다.
- Null은 반드시 빈 값으로 해석되어야 합니다.
- 문자열 유형은 반드시 문자열로 해석되어야 하며, 이 값에 대한 속성의 정의에 따라 URI, 데이터 스탬프 또는 기타 값과 같은 보다 구체적인 데이터 유형으로 추가 구문 분석될 수 있습니다.

JSON을 생성하는 구현자는 알고리즘이 [INFRA] 사양의 JSON 사용 규칙과 일치하는지 확인하는

것이 좋습니다.

@context 이름은 JSON-LD 예약어이므로 사용해서는 안 됩니다.

(2) JSON-LD

[JSON-LD]는 링크드 데이터(Linked Data)를 직렬화하는 데 사용되는 JSON 기반 형식입니다.

(분해자 메타 데이터에 application/did+ld+json의 content-type으로 지정된 것과 같이) JSON-LD로 DID 문서를 생성하고 소비하는 경우, 다음과 같은 규칙을 반드시 따라야 합니다.

- @id 및 @type키워드는 id 및 type의 별칭이며, 개발자는 이 사양을 관용적으로 JSON에서 사용할 수 있습니다..
- JSON-LD는 노드 식별자로서 IRI를 허용하지만, DID 문서는 단지 DIDs를 설명하는 데 명시적으로 제한됩니다. DID 객체에 적용된 Id 값은 반드시 DID에 유효해야 하며, 다른 종류의 IRI가 아닙니다.
- 정수, 날짜, 측정 단위 및 URL과 같은 데이터 유형은 이를 필요로 하는 사용 사례에 대해 유형을 보장하기 위해 자동으로 입력됩니다.

① 생산

DID 문서는 JSON 프로세서에 다음과 같이 배열됩니다: DID 문서는 반드시 @context 속성을 포함합니다.

@context

@context 속성값은 반드시 하나 이상의 URI이어야 합니다. 추가적으로, 첫 번째 URI값은 https://www.w3.org/ns/did/v1입니다. @context 속성 모든 멤버는 서로 다른 표현에서 상호 연동을 위해 DID 스펙 레지스트리에 존재해야 합니다. 멤버가 DID 스펙 레지스트리에 없는 경우 DID 문서는 표현 간에 상호 연동되지 않습니다.

② 소비

최상위 요소는 반드시 JSON 개체이어야 합니다. 최상위 레벨의 다른 데이터 유형은 오류이며 반드시 거부되어야 합니다. 이 최상위 JSON 개체는 정의된 @context필드 규칙에 따라 JSON-LD 프로세스를 사용하여 해석됩니다.

@context

@context 속성값은 반드시 하나 이상의 URI이어야 합니다.

첫 번째 URI 값은 https://www.w3.org/ns/did/v1입니다. 하나 이상의 URI가 제공되는 경우, URI는 반드시 정렬된 세트로 해석될 수 있습니다. 컨텍스트에 대한 기계 판독 가능 정보를 포함하는 문서의 결과에 따라 각 URL을 역참조할 수 있도록 권장합니다. 다른 표현과의 상호 연동을 위하여 URL은 JSON-LD 컨텍스트를 참조하는 DID 사양 레지스트리[DID-SPEC-REGISTRIES]에 등록되고, JSON-LD 컨텍스트 콘텐츠의 암호화 해시와 연결되어야 합니다. 이는 JSON-LD 소비자의 정보 해석이 DID 사양 레지스트리[DID-SPEC-REGISTRIES]에 의존하는 다른 소비자의 다른 표현에 대한 해석과 동일하도록 보장합니다.

알 수 없는 객체 멤버 이름은 반드시 알 수 없는 속성으로 무시되어야 합니다.

(3) CBOR

JSON(Javascript Object Notation) [RFC8259]와 마찬가지로 CBOR(Concise Binary Object Representation) [RFC7049]은 구조화된 데이터의 연동 가능한 표현을 위한 형식화 규칙 세트를 정의합니다. CBOR은 보다 간결하고 기계가 읽을 수 있는 언어 독립적인 데이터 교환 형식으로 자체 설명이 가능하며 상호 연동을 위해 내부에 데이트의 의미를 가지고 있습니다. 특정 제약 조건을 통해 CBOR은 DID 문서 모델(데이터 모델 및 DID 문서에 설명됨)과 기타 핵심 표현 간의 변환을 위해 모든 JSON 데이터 유형(JSON-LD 포함)을 지원할 수 있습니다.

CDDL(Concise Data Definition Language) [RFC8610]은 CBOR(Concise Binary Object Representation) 및 확장된 JSON 데이터 구조로 표현하는 데 사용되는 표기법입니다. 다음 표기법은 다른 핵심 표현 간의 결정론적 매핑에 대한 특정 제약을 사용하여 CBOR 표현의 DID 문서 모델을 표현합니다.

Example 22 : CDDL 표기법으로 표현된 CBOR 용 DID 문서 데이터 모델

```
DID- document = {
  ? @context : uri
  id : did
  ? publicKey : [* publicKey]
  ? authenticaion : [* did // * publicKey // * tstr]
  ? service: [+ service]
```

```
  ? controller : did / [* did]
  ? created : time
  ? updated : time
  proof : any
}

publicKey = {
  id : did
  type : text
  controller : uri
}

did = tstr .pcre "^ did \\ : (? <method-name> [a-z0-9] {2,}) \\ : (? <method-specific-id>
[A-Za-z0-9 \\. \\-\\ : \\ _] +) "

did-url = tstr .pcre "^ did \\ : (? <method-name> [a-z0-9] {2,}) \\ : (? <method-specific-id>
[A-Za-z0 -9 \\. \\-\\ : \\ _] +) \\; (? <path> [A-Za-z0-9 \\ /) (? <query> \\? [a-z0- 9
\\ = \\ &]) # (? <fragment>. +) "

service = {
  id : did-url
  type : text
  serviceEndpoint : uri
  ? description : text
  * tstr => any
}
```

① 생산

CBOR로 표현된 DID 문서를 생성할 때, 결정적 맵핑을 위하여 CBOR[RFC7049] 사양 섹션 3.9에 나온 제안에 추가적으로 DID 문서의 제약들은 반드시 다음을 따라야 합니다:

- 맵 키는 반드시 문자열이어야 합니다.
- 정수 인코딩은 반드시 가능한 한 짧아야 합니다.
- CBOR 주요 유형 2-5에 나오는 길이 표현은 반드시 가능한한 짧아야 합니다.
- 모든 부동 소수점 값은 반드시 정수 값일 경우에도 64비트로 인코딩되어야 합니다.

Example 21 : CBOR 로 표시되고 쉽게 읽을 수 있도록 진단 주석 모드로 내보낸 DID 문서의 예

```
a7                                                  #map (7)
62                                                  # text (2)
   6964                                             #    "id"
78 40                                               # text (64)
   6469643a6578616d706c653a31324433                 #    "did : example : 12D3"
   4b6f6f574d4864727a6377706a626472                 # "KooWMHdrzcwpjbdr"
   5a7335474771455241766367715833362                # "Zs5GGqERAvcgqX3b"
   3564707550745061396f743639796577                 # "5dpuPtPa9ot69yew"
65                                                  # text (5)
   70726f6f66                                       #    "proof"
a4                                                  # map (4)
   64                                               #    text (4)
      74797065                                      #       "type"
   74                                               #    text (20)
      65643235353139353696e617475726532303138       #       "ed25519Signature2018"
   67                                               #    text (7)
      63726561746564                                #       "created"
   74                                               #    text (20)
      323032302d30352d30315430333a30303a30325a       # "2020-05-01T03 : 00 : 02Z"
   67                                               #    text (7)
      63726561746f72                                #       "creator"
```

```
   78 8c                                                  #    text (140)
      6469643a6578616d706c653a31324433                    #      "did : example : 12D3"
      4b6f6f574d4864727a6377706a626472                    #      "KooWMHdrzcwpjbdr"
      5a7335474771455241766367715836 2                    #      "Zs5GGqERAvcgqX3b"
      3564707550745061396f743639796577                    #      "5dpuPtPa9ot69yew"
      3b206578616d706c653a6b65793d6964                    #    "; example : key = id"
      3d626166797265696375627478357771                    #    "= bafyreicubtx5wq"
      6f336e6f73633463617a726b63746668                    #    "o3nosc4cazrkctfh"
      776436726577657a6770776f65347377                    #    "wd6rewezgpwoe4sw"
      69726c733465626468733269                            #    "irls4ebdhs2i"
   6e                                                     #  text (14)
      7369676e617475726556616c7565                        #    "signatureValue"
   78 58                                                  #  text (88)
      6f3972364c78676f474e38466f616565                    #    "o9r6LxgoGN8Foaee"
      5541364564446376313247764 47a4645                   #    "UA6EdDcv12GvDzFE"
      6d43676a577a76707572325953517941                    #    "mCgjWzvpur2YSQyA"
      3857327230535357554b2b6e4835744d                    #    "8W2r0SSWUK + nH5tM"
      717a61464c756e3677775a31456f7433                    #    "qzaFLun6wwZ1Eot3"
      37616d4744673d3d                                    #    "7 amGDg=="
67                                                        #  text (7)
   63726561746564                                         #    "created"
74                                                        #  text (20)
   323031382d31322d30315430333a30303a30305a               #   "2018-12-01T03 : 00 : 00Z"
67                                                        #   text (7)
   75706461746564                                         #    "updated"
74                                                        #  text (20)
   323032302d30352d30315430333a30303a30305a               #   "2020-05-01T03 : 00 : 00Z"
68                                                        #  text (8)
   40636f6e74657874                                       #    "@context"
78 1c                                                     #  text (28)
```

```
   68747470733a2f2f7777772e77332e6f                  #       "https : //www.w3.o"
   72672f6e732f6469642f7631                          #       "rg / ns / did / v1"
69                                                   #   text (9)
   7075626c69634b6579                                #     "verificationMethod"
81                                                   #   array (1)
   a5                                                #     map (5)
      62                                             #       text(2)
         6964                                        #         "id"
      78 85                                          #       text (133)
         626166797265696375627478357771 6f           #         "bafyreicubtx5wqo"
         336e6f73633463617a726b6374666877            #         "3nosc4cazrkctfhw"
         6436726577657a6770776f6534737769            #         "d6rewezgpwoe4swi"
         726c733465626468733269 3b6578616d            #         "rls4ebdhs2i; exam"
         706c653a6b65793d6964626166797265            #         "ple : key = idbafyre"
         6963756274783577716f336e6f736334            #         "icubtx5wqo3nosc4"
         63617a726b6374666877643672657765            #         "cazrkctfhwd6rewe"
         7a6770776f6534737769726c73346562            #         "zgpwoe4swirls4eb"
         6468733269                                  #         "dhs2i"
      64                                             #       text (4)
         74797065                                    #         "type"
      6e                                             #       text(14)
         45644473615075626c69634b6579                #         "EdDsaPublicKey"
      65                                             #       text(5)
         6375727665                                  #         "curve"
      67                                             #       text (7)
         65643235353139                              #         "ed25519"
      67                                             #       text (7)
         65787069726573                              #         "expires"
      74                                             #       text(20)
         323031392d31322d30315430333a30303a30305a    #       "2019-12-01T03 : 00 : 00Z"
```

```
      6f                                                  #     text (15)
         7075626c69634b6579426173653634                   #       "publicKeyBase64"
      78 2c                                               #     text (44)
         716d7a3774704c4e4b4b4b646c376344                 #       "qmz7tpLNKKKdl7cD"
         375062656a4469425670374f4e706d5a                 #       "7PbejDiBVp7ONpmZ"
         62666d633763454b396d673d                         #     "bfmc7cEK9mg ="
6e                                                       #   text(14)
   61757468656e7469636174696f6e                          #     "authentication"
81                                                       #   array(1)
   78 83                                                 #     text (131)
      6469643a6578616d706c653a31324433                   #       "did : example : 12D3"
      4b6f6f574d4864727a6377706a626472                   #       "KooWMHdrzcwpjbdr"
      5a7335474771455241766367715833362                  #       "Zs5GGqERAvcgqX3b"
      3564707550745061396f743639796577                   #       "5dpuPtPa9ot69yew"
      3b6b65792d69643d6261667972656963                   #       "; key-id = bafyreic"
      756274783577716f336e6f7363346361                   #       "ubtx5wqo3nosc4ca"
      7a726b6374666877643672657765 7a67                  #       "zrkctfhwd6rewezg"
      70776f6534737769726c733465626468                   #       "pwoe4swirls4ebdh"
      733269                                             #     "s2i"
```

② 소비

CBOR로 표현되는 DID 문서를 소비하는 경우 추가적으로 DID 문서 모델의 제한 조건에 따라 결정적인 맵핑을 하도록 CBOR [RFC7049] 사양이 제한하는 것과 함께 반드시 다음을 따라야 합니다.

- 모든 맵의 키는 가장 낮은 값에서 가장 높은 값으로 정렬되어야 합니다. 정렬은 키 표현의 바이트에서 수행됩니다.
- 무한 길이 항목은 유한 길이 항목으로 만들어야 합니다.

③ CBOR 확장성

CBOR에서 확장성의 한 시작점은 CBOR 태그를 사용하는 것입니다. [RFC7049]는 CBOR 태그 레지스트리를 통해 지원되는 데이터 유형 세트를 확장할 수 있는 태그 지정 메커니즘뿐만 아니라 기본 데이터 유형 세트를 정의합니다. 이를 통해 태그는 뒤에 오는 데이터의 의미론적 설명을 향상시킬 수 있습니다.

ⓐ DagCBOR

DagCBOR은 추가 제약과 함께 위에 언급한 대로 CBOR 인코딩한 Directed Acyclic Graph 모델로 DID 문서를 표현하기 위한 CBOR의 더 제한된 서브 세트입니다. DagCBOR은 주어진 객체를 인코딩하는 단일 방법이 있어야 하며 인코딩된 양식에는 왕복 디코딩/인코딩에서 무시되거나 손실될 수 있는 불필요한 데이터가 포함되어 있지 않아야 합니다. DagCBOR로 표현된 DID 문서는 다음과 같은 규칙을 반드시 따라야 합니다.

- CID 태그(42) 이외의 CBOR 태그를 사용하지 마십시오.

Example 24 : DagCBOR 로 DID 문서를 전 예제와 동일하지만 쉽게 읽을 수 있도록 JSON 으로 직렬화

```
{ "@context": "https://www.w3.org/ns/did/v1",
  "authentication": [
    "did : example : 12D3KooWMHdrzcwpjbdrZs5GGqERAvcgqX3b5dpuPtPa9ot69yew;
key-id = bafyreicubtx5wqo3nosc4cazrkctfhwd6rewezgpwoe4swirls4ebdhs2i"
  ],
  "created": "2018-12-01T03 : 00 : 00Z",
  "id": "did : example : 12D3KooWMHdrzcwpjbdrZs5GGqERAvcgqX3b5dpuPtPa9ot69yew",
  "proof": {
    "created": "2020-05-01T03 : 00 : 02Z",
    "creator": "did : example :
12D3KooWMHdrzcwpjbdrZs5GGqERAvcgqX3b5dpuPtPa9ot69yew; example : key
= id = bafyreicubtx5wqo3nosc4cazrkctfhwd6rewezgpwoe4swirls4ebdhs2i",
    "signatureValue":
"o9r6LxgoGN8FoaeeUA6EdDcv12GvDzFEmCgjWzvpur2YSQyA8W2r0SSWUK +
nH5tMqzaFLun6wwZ1Eot37amGDg ==",
```

```
      "type": "ed25519Signature2018"
    },
    "verificationMethod": [
      {
        "curve": "ed25519",
        "expires": "2019-12-01T03 : 00 : 00Z",
        "id": "bafyreicubtx5wqo3nosc4cazrkctfhwd6rewezgpwoe4swirls4ebdhs2i;
example : key =
idbafyreicubtx5wqo3nosc4cazrkctfhwd6rewezgpwoe4swirls4ebdhs2i",
        "publicKeyBase64":
"qmz7tpLNKKKdl7cD7PbejDiBVp7ONpmZbfmc7cEK9mg =",
        "type": "EdDsaPublicKey"
      }
    ],
    "updated": "2020-05-01T03 : 00 : 00Z"
}
```

ⓑ COSE 서명

DID 문서 증명은 CBOR Object Signing and Encryption(COSE) [RFC8152]를 위한 태그 98과 같은 CBOR 시맨틱 태깅을 사용하여 구성될 수 있습니다.

Example 25 : 진단 주석 형식으로 표현된 태그 98 및 42 를 사용한 CBOR 의 COSE 시그니처 확장성 예

```
D8 62                                                    # tag (98)
67                                                       #    text (7)
   7061796c6f6164                                        #       "payload"
d8 2a                                                    #    tag (42)
   58 25                                                 #       bytes (37)
      000171122206c8fdc5c3d2302dda95034  #"\x0  0\x01q\x12 l\x8f\xdc\\= #\x02\xdd\xa9P4"
      f9de57a8591918ecb7d7789387c547f7  # "\xf9\xdeW\xa8Y\x19\x18\xec\xb7\xd7x\x93\x87\
a89d05e72f                                # "\xa8\x9d\x05\xe7/"
```

```
69                                                  #   text (9)
   70726f746563746564                               #    "protected"
a0                                                  #   map (0)
6a                                                  #   text (10)
   7369676e617475726573                             #    "signatures"
81                                                  #  array (1)
   a3                                               #   map (3)
      69                                            #    text (9)
         70726f746563746564                         #     "protected"
      66                                            #    text (6)
         613130313236                               #     "a10126"
      69                                            #    text (9)
         7369676e6174757265                         #     "signature"
      78 80                                         #   text (128)
         653261656166643430643963964313964          #    "e2aeafd40d69d19d"
         666536653532303737633564376666 34          #    "fe6e52077c5d7ff4"
         653430383238326362656662356430 36          #    "e408282cbefb5d06"
         636266343134616632653139643938 32          #    "cbf414af2e19d982"
         616334356163393862383534346339 30          #    "ac45ac98b8544c90"
         386234353037646531653930623731 37          #    "8b4507de1e90b717"
         633364333438313666653932366132 62          #   "c3d34816fe926a2b"
         393866353361666432666130663330 61          #   "98f53afd2fa0f30a"
      6b                                            #   text (11)
         756e70726f746563746564                     #    "unprotected"
      a1                                            #   map (1)
         63                                         #    text (3)
            6b6964                                  #     "kid"
         78 85                                      #    text (133)
            6469643a697069643a313244334b6f6f        #     "did : ipid : 12D3Koo"
            574d4864727a6377706a6264725a7335        #     "WMHdrzcwpjbdrZs5"
```

```
                474771455241766367715833623564706e  #       "GGqERAvcgqX3b5dp"
                7550745061396f7436397965773b6970    #       "uPtPa9ot69yew; ip"
                69643a6b65792d69643d626166797265    #     "id : key-id = bafyre"
                6963756274783577716f336e6f736334    #     "icubtx5wqo3nosc4"
                63617a726b6374666877643672657765    #     "cazrkctfhwd6rewe"
                7a6770776f6534737769726c73346562    #     "zgpwoe4swirls4eb"
                6468733269                          #     "dhs2i"
6b                                                  # text (11)
   756e70726f746563746564                           #   "unprotected"
a0                                                  # tag (0)
```

7) 메소드(Method)

DID 메소드는 다양한 검증 가능한 데이터 저장소에서 본 스펙을 구현하는 수단을 제공합니다. 새로운 DID 메소드는 자체 사양에 정의되어 있으므로 동일한 DID 메소드의 서로 다른 구현 간의 상호 연동이 보장됩니다. 이 섹션에서는 DID 메소드의 관련 사양에 의해 충족되는 모든 DID 메소드에 대한 요구 사항을 정의합니다.

특정 DID 메소드에 특정한 DID 문서에 속성을 추가 하려면 § 4.3 확장성을 참조하십시오.

(1) 메소드 스키마(Methods schemes)

DID 메소드 스펙에서는 반드시 정확히 하나의 메소드 이름에 의해 식별된 하나의 특정 메소드 DID 스키마를 정의해야 합니다(§ 3.1 DID 구문의 method-name 규칙을 참조하십시오).

새로운 DID 메소드 사양의 작성자는 발행 당시에 알려진 모든 DID 메소드 이름 중에서 고유한 메소드 이름을 사용해야 합니다.

메소드 이름은 다섯 자 이하로 해야 합니다.

참고 : 고유한 DID 메소드 이름

DID 메소드 이름을 할당하거나 승인하는 중앙 권한이 없기 때문에, 특정 DID 메소드 이름이 고유한지 여부를 확인할 방법이 없습니다. 이 문제를 해결하기 위해 알려진 DID 메소드 이름 및 관련 사양의 비공식 목록이 [DID-SPEC-REGISTRIES]의 일부인 DID 메소드 레지스트리에서 관리되고 있습니다.
새로운 DID 메소드 사양의 작성자는 다른 구현자와 커뮤니티 구성원이 기존 DID 메소드 개요를 볼 수 있도록 DID 메소드 레지스트리에 메소드 이름을 추가하는 것이 좋습니다.

DID 메소드의 사양은 반드시 DID의 method-specific-id 구성 요소를 생성하는 방법을 지정해야 합니다.

DID 메소드 사양은 DID의 method-specific-id 컴포넌트가 어떻게 생성되는지를 반드시 지정해야 합니다.

method-specific-id 값은 자체로 글로벌하게 고유한 값을 가집니다. 메소드에 의해 생성된 모든 DID는 글로벌적으로 고유해야 합니다.

필요한 경우, 메소드별 DID 스키마는 여러 개의 method-specific-id 형식을 정의할 수도 있습니다. 하나의 메소드별 DID스키마는 가능한 적은 개수의 method-specific-id 형식을 정의할 것을 권장합니다.

method-specific-id 형식은 콜론을 포함합니다. 콜론의 사용은 반드시 method-specific-id ABNF 규칙을 구문적으로 준수해야 합니다.

참고 : 메소드별 ID의 콜론

method-specific-id에서 콜론의 의미는 전적으로 메소드에 따라 다릅니다. 개발자는 일반적으로 모든 DID 메소드들에 적용할 수 있는 콜론과 관련된 의미나 동작도 가정하지 않는 것이 좋습니다.

(2) 메소드 작업(Method Operations)

이 섹션에서는 DID 문서에서 수행할 수 있는 작업과 관련하여 DID 메소드 사양에 대한 요구사항을 설명합니다.

작업을 수행할 당사자의 권한을 결정하는 것은 메소드에 따라 다릅니다. 예를 들어, DID 메소드는 다음과 같습니다.

- controller 속성을 사용하도록 합니다.
- 업데이트/비활성화 작업이 허용되는지 여부를 결정하는 데 authentication에 나열된 검증 방법을 사용합니다.
- capabilityInvocation에 지정된 검증 방법을 사용하여 DID 문서를 업데이트하는 기능의 호출을 확인할 수 있는지를 결정하는 것과 같은 경우를 위해 DID 문서의 다른 구조를 사용합니다.
- 이를 결정하기 위해 DID 문서를 전혀 사용하지 않고 메소드에 "내장된" 규칙을 가지고 있습니다.

각 DID 메소드는 반드시 필요한 암호화 작업을 포함하여 권한 부여를 어떻게 구현할지 정의해야 합니다.

① 생성

DID 메소드 사양은 반드시 통제된 증명을 생성하는 데 필수적인 암호화 작업을 포함하여 DID 컨트롤러가 어떻게 DID와 해당 DID와 관련있는 DID 문서를 검증 가능한 데이터 저장소에 저장해야 하는지를 정의해야 합니다.

② 읽기/확인

DID 메소드 사양은 반드시 DID 해석자(DID resolver)가 어떻게 응답 인증을 검증하는지를 포함하여, 어떻게 DID 해석자가 검증 가능한 데이터 저장소로부터 DID 문서를 DID를 통해 요청하도록 하는지에 대한 부분을 정의해야 합니다.

③ 업데이트

DID 메소드 사양은 반드시 제어를 할 수 있는 증거를 확립하고 업데이트가 가능하지 않은 상태인지에 필요한 모든 암호화 작업을 포함하여 클라이언트가 검증 가능한 데이터 저장소에 DID 문서를 업데이트할 수 있는 방법을 정의해야 합니다.

DID에 대한 업데이트는 생성 후 DID 문서를 생성하는 데 사용되는 데이터에 대한 변경 사항입니다. DID 메소드 개발자는 업데이트를 구성하는 항목과 지정된 DID 메소드가 지원하는 DID 문서의 속성을 정의할 책임이 있습니다. 예를 들어 키 자료를 변경하지 않는 업데이트 작업은 DID 문서를 변경하지 않지만 유효한 업데이트일 수 있습니다.

④ 비활성화

DID 메소드 사양은 반드시 클라이언트가 검증 가능한 데이터 저장소에 DID를 비활성화하는 방법을 정의해야 합니다. 여기에는 비활성화시키는 증거를 확립하거나 비활성화가 가능하지 않는 상태인지를 알기 위해 필요한 모든 암호화 작업이 포함됩니다.

(3) 보안 요구 사항

DID 메소드 사양은 반드시 자신의 보안 고려 사항 섹션을 포함해야 합니다. 이 섹션은 반드시 사양에 정의 된 DID 작업에 대해 [RFC3552]의 섹션 5에 언급된 모든 요구 사항을 고려해야 합니다 .

적어도 도청, 재생, 메시지 삽입, 삭제, 수정 및 중간자(man-in-the-middle)같은 공격형태를 반드시 고려해야 합니다. 잠재적인 서비스 거부 공격도 반드시 식별해야 합니다.

이 섹션에서는 [RFC3552]의 섹션 5에 따라 위협 완화가 배포된 후 잔류 위험(예 : 관련 프로토콜의 손상, 잘못된 구현 또는 암호)에 대해 논의해야 합니다.

이 섹션은 § 7.2 방법 작업에서 요구하는 모든 작업에 대해 무결성 보호 및 업데이트 인증을 제공 해야 합니다.

기술이 인증을 포함하는 경우, 특히 사용자 호스트 인증이 포함된 경우 인증 방법의 보안을 명확하게 지정해야 합니다.

DID 메소드는 반드시 DID가 고유하게 할당되는 것으로 입증된 정책 메커니즘을 논의해야 합니다. DID는 [RFC8141]에 정의된 URN의 기능적 정의에 부합해야 합니다. 즉, DID는 리소스에 한번 할당되면 다른 리소스에 다시 할당되지 않는 영구 식별자입니다. 이는 특정 허가 권한 세트에 따라 특정 당사자를 식별하는 데 DID를 사용할 수 있으므로 보안 컨텍스트에서 특히 중요합니다.

메소드별 엔드 포인트 인증도 반드시 논의되어야 합니다. DID 메소드에 네트워크 토폴로지(topology) 변화와 함께 DLTS를 이용하는 경우, 때로는 필요한 컴퓨팅 리소스를 줄이기 위해 light nod 또는 씬 클라이언트로 구현되기도 합니다. DID 메소드 구현에 사용할 수 있는 토폴로지의 보안 가정은 반드시 논의되어야 합니다.

프로토콜이 암호화 보호 메커니즘을 통합하는 경우, DID 메소드 사양은 반드시 데이터의 어떤 부분이 보호되고 보호가 무엇인지 명확하게 나타내야 하며 암호화 보호가 취약한 유형의 공격에 대한 지표를 제공해야 합니다. 예를 들어 인터그리티, 기밀성, 엔드 포인트 인증 등이 있습니다.

비밀로 유지될 데이터(키 자료, 무작위 시드 등)는 명확하게 표시되어야 합니다.

DID 메소드는 모든 알려진 DLTs와 같은 피어-투-피어 컴퓨팅 리소스를 사용하는 경우, 해당 자원의 예상 부담 및 서비스 거부와 관련하여 논의되어야 합니다.

새로운 인증 서비스 엔드 포인트 유형을 도입하는 DID 메소드(§ 5.5 서비스 엔드 포인트 참조)는 지원되는 인증 프로토콜의 보안 요구 사항을 고려해야 합니다.

(4) 개인정보 요구 사항

DID 메소드 사양은 반드시 § 10. 개인정보 고려 사항을 가리키는 경우에만 자체 개인정보 고려 사항 섹션을 포함해야 합니다.

DID 메소드 사양의 개인정보보호 고려 사항 섹션은 반드시 메소드 별 형식으로 적용할 수 있는 [RFC6973]의 섹션5의 하위 섹션에서 논의되어야 합니다. 고려해야 할 하위 섹션은 감시(surveillance), 저장된 데이터 손상, 원치 않는 트래픽, 오인(misattribution), 상관관계(correlation), 식별, 2차 사용, 공개(disclosure), 제외(exclusion)입니다.

8) 분해(Resolution)

이 섹션에서는 DID 분해 및 DID URL 역참조의 입력 및 출력을 정의합니다. 이러한 기능은 추상적인 방식으로 정의됩니다. 정확한 구현은 이 사양의 범위를 벗어나지만 개발자에 대한 몇 가지 고려 사항은 [DID-RESOLUTION]에서 설명합니다.

모든 호환 DID 분해자(resolver)는 반드시 적어도 하나의 DID 메소드에 대한 DID 분해 기능이 구현되어야 하며, 반드시 적어도 하나의 호환 표현으로 DID 문서로 돌아갈 수 있도록 해야 합니다.

(1) DID 분해(Resolution)

DID 분해 기능은 적용 가능한 DID 메소드의 "읽기" 작업을 사용하여 DID 문서로 DID를 확인하는 것입니다(§ 7.2.2 읽기/확인 참조). 이 프로세스가 어떻게 수행되는지에 대한 세부 사항은 이 사양의 범위를 벗어나지만 모든 호환되는 구현은 반드시 다음과 같은 추상 형식을 가진 두 가지 기능을 구현해야 합니다.

참고

```
resolve (did, did-resolution-input-metadata)
    -> (did-resolution-metadata, did-document, did-document-metadata)
resolveStream (did, did-resolution-input-metadata)
    -> (did-resolution-metadata, did-document-stream, did-document-metadata)
```

이러한 함수의 입력 변수는 다음과 같아야 합니다.

did

하나의 문자열로 구성된 적합한 DID. 이것은 분해를 위한 DID입니다. 이 입력은 필수적입니다.

did-resolution-input-metadata

did에 추가적으로 resolve 및 resolveStream 함수에 입력 옵션으로 구성된 메타 데이터 구조입니다. 이 사양에 정의된 속성은 § 8.1.1 DID 분해 입력 메타 데이터 속성에 있습니다. 입력값은 필수이지만, 구조는 비어 있을 수도 있습니다.

이러한 함수의 출력변수는 다음과 같아야 합니다.

did-resolution-metadata

DID 분해 프로세스의 결과와 관련된 값들로 구성된 메타 데이터 구조입니다. 이 구조는 필수적이고 비워져서는 안 됩니다. 이 메타 데이터는 일반적으로 분해 프로세스 자체에 대한 데이터를 나타내기 때문에 resolve 및 resolveStream 함수 호출 간에 변경됩니다. 이 사양에 정의된 속성은 § 8.1.2 DID 분해 메타 데이터 속성에 있습니다. 분해가 성공하고, resolveStream 함수가 호출된 경우 이 구조는 반드시 결과에 typedid-document-stream의 mime 유형을 포함하는 content-type 속성을 포함해야 합니다. 분해가 실패하면, 이 구조는 반드시 오류를 설명하는 error 속성을 포함해야 합니다.

did-document

DID 분해가 성공한 경우와 resolve 함수가 호출되는 경우, 이는 반드시 추상적 데이터 모델에 부합하는 DID 문서여야 합니다. 분해에 실패하면 이 값은 반드시 비어있어야 합니다.

did-document-stream

DID 분해가 성공하는 경우와 resolveStream 함수가 호출되는 경우에는 반드시 준수된 표현의 하나로 분해된 DID 문서의 바이트 스트림이 있어야 합니다. 바이트 스트림은 다시 확인하고 처리할 수 있는 DID 추상 데이터 모델로 resolveStream 함수를 호출하는 호출자로 분석될 수 있습니다. 분해가 실패하면 이 값은 반드시 빈 스트림이어야 합니다.

did-document-metadata

분해가 성공하면, 이것은 반드시 메타 데이터 구조여야 합니다. 이 구조에는 did-document 또는 did-document-stream에 포함된 DID 문서에 대한 메타 데이터가 포함됩니다. 이 메타 데이터는 DID 문서 변경 없이는 resolve 함수의 호출 간에 변경이 없습니다. DID 문서에 대한 데이터를 나타내는 것입니다. 분해에 실패하면 이 출력은 반드시 빈 메타 데이터 구조여야 합니다. 이 사양에 정의된 속성은 § 8.1.3 DID 문서 메타 데이터 속성에 있습니다.

DID 분해자(DID resolver) 구현은 어떤 식으로든 이러한 함수의 서명을 변경해서는 안 됩니다. DID 분해자 구현은 DID 분해 프로세스를 시행하도록 하는 메소드 별 내부 함수에 resolve와 resolveStream 함수를 매핑할 수도 있습니다. DID 분해자 구현은 여기에 지정된 resolve 함수에 추가적으로 다른 서명을 가진 추가 기능을 구현하고 노출할 수도 있습니다.

① DID 분해 입력 메타 데이터 속성

이 구조 내에서 가능한 속성과 가능한 값은 [DID-SPEC-REGISTRIES]에 의해 정의됩니다. 이 사양은 다음과 같은 공통 속성을 정의합니다.

accept

호출자가 선호하는 DID 문서 표현의 MIME 유형입니다. DID 분해자를 구현하는 데, 반환되는 did-document-stream에 포함된 표현을 결정하는 데 이러한 표현이 지원되거나 이용 가능한 경우 이 값을 사용할 수 있습니다. 이 속성은 옵션입니다. resolveStream 함수가 호출된 경우에만 사용되며 resolve 함수가 호출되면 무시해야 합니다.

② DID 확인 메타 데이터 속성

이 구조 내에서 가능한 속성과 가능한 값은 [DID-SPEC-REGISTRIES]에 의해 정의됩니다. 이 사양은 다음과 같은 공통 속성을 정의합니다.

content-type

반환된 did-document-stream의 MIME 유형입니다. 분해가 성공하고 resolveStream 함수가 호출된 경우 이 속성이 요구됩니다. resolve 함수가 호출된 경우 반드시 이 속성이 존재하면 안됩니다. 이 속성의 값은 반드시 준수된 표현의 하나인 MIME 타입이어야 합니다. resolveStream 함수 호출자는 반드시 DID 문서 추상 데이터 모델에 이 함수로 반환되는 did-document-stream을 구문 분석하고 처리하는 방법을 결정할 때 이 값을 사용해야 합니다.

error

분해 프로세스의 오류 코드입니다. 분해 프로세스에 오류가 있는 경우 이 속성은 필수입니다. 이 속성의 값은 단일 키워드 문자열입니다. 이 필드의 가능한 속성 값은 [DID-SPEC-REGISTRIES]에 의해 정의됩니다. 이 사양은 다음 오류 값을 정의합니다.

Invalid-did

DID 분해 함수에 지원되는 DID가 유효한 구문과 일치하지 않습니다.(§ 3.1 DID 구문 참조.)

unauthorized

호출자는 이 DID 분해자로 해당 DID를 확인할 권한이 없습니다.

not-found

이 DID 분해자는 확인 요청의 결과로 DID 문서를 반환할 수 없습니다.

③ DID 문서 메타 데이터 속성

이 구조 내에서 가능한 속성과 가능한 값은 [DID-SPEC-REGISTRIES]에 의해 정의됩니다. 이 사양은 다음과 같은 공통 속성을 정의합니다.

created

DID 문서 메타 데이터는 생성작업의 타임 스탬프를 지시하는 created 속성를 포함해야 합니다. 이 속성은 주어진 DID 메소드를 지원하지 않을 수도 있습니다. 속성 값은 반드시 W3C XML Schema Definition Language(XSD) 1.1 Part 2: Datatypes[XMLSCHEMA11-2 섹션 3.3.7에 정의된 유효한 XML 날짜 값이어야 합니다. 이 날짜 시간 값은 반드시 뒤에 "Z"로 표시된 대로 UTC 00:00으로 정규화되어야 합니다.

updated

DID 문서 메타 데이터는 최종 업데이트 작업의 타임 스탬프를 나타내는 updated 속성을 포함해야 합니다. 이 속성은 주어진 DID 방법을 지원하지 않을 수도 있습니다. 속성값은 반드시 created 속성과 같은 포맷 규칙에 따라야 합니다.

(2) DID URL 역참조

DID URL 참조 기능은 DID URL을 DID 메소드, 메소드 특정 식별자, 경로, 쿼리와 프래그먼트를 포함하는 DID URL의 컴포넌트에 의존하는 콘텐츠를 가진 리소스로 역참조합니다. 이 과정은 DID URL을 속에 포함되어 있는 DID의 DID 분해자에 의존합니다. 이 과정이 어떻게 진행되는지 자세한 사항은 이 문서의 범위에서 벗어나 있습니다만, 모든 준수된 구현체는 다음의 추상화된 양식을 가진 기능을 구현해야 합니다.

참고

```
dereference (did-url, did-url-dereferencing-input-metadata)
        -> (did-url-dereferencing-metadata, content-stream, content-metadata)
```

(3) 메타 데이터 구조

입력 및 출력 메타 데이터는 종종 DID 확인, DID URL 역참조 및 기타 DID 관련 프로세스 중에 포함됩니다. 이 메타 데이터가 통신하는 데 사용되는 구조는 반드시 속성들의 맵입니다. 각 속성 이름은 반드시 문자열이어야 합니다. 각 속성 값은 반드시 문자열, 맵, 목록, boolean 또는 null이어야 합니다. 맵과 목록 등 복잡한 데이터 구조 내의 값은 반드시 이러한 데이터 유형 중 하나여야 합니다. 모든 메타 데이터 속성 정의는 반드시 해당 값에 대한 추가 형식 또는 제한을 포함하여 값 유형을 정의해야 합니다(예 : 날짜 또는 십진 정수 형식의 문자열). 속성 정의는 가능한 값에 대한 문자열을 사용하는 것이 권장됩니다.

하나의 입력 또는 출력으로 메타 데이터 구조를 사용하여 구현하는 모든 기능은 반드시 결정적 방식으로 설명되는 모든 데이터 유형들을 완전히 나타낼 수 있어야 합니다. 메타 데이터 구조를 사용하는 입력 및 출력은 직렬화가 아닌 데이터 유형으로 정의되므로 표현 방법은 함수 구현 내부에 있으며 이 사양의 범위를 벗어납니다.

다음 예는 DID 확인 입력 메타 데이터로 사용될 수 있는 JSON 인코딩 메타 데이터 구조를 보여줍니다.

Example 26 : JSON 인코딩된 DID 확인 입력 메타 데이터 예제

```
{
  "accept": "application/did+ld+json"
}
```

이 예는 다음 형식의 메타 데이터 구조에 해당합니다.

Example 27 : DID 해상도 입력 메타 데이터 예

```
«[
  "accept" → "application/did+ld+json"
]»
```

다음 예제는 DID를 찾을 수 없는 경우, DID 확인 메타 데이터로 사용할 수 있는 JSON 인코딩 메타 데이터 구조를 보여줍니다.

Example 28 : JSON 인코딩된 DID 확인 메타 데이터 예제

```
{
  "error": "not-found"
}
```

이 예는 다음 형식의 메타 데이터 구조에 해당합니다.

Example 29 : DID 확인 메타 데이터 예

```
«[
  "error" → "not-found"
]»
```

다음 예제는 DID 문서와 관련된 타임 스탬프를 설명하기 위해 DID 문서 메타 데이터로 사용될 수 있는 JSON 인코딩된 메타 데이터 구조를 보여줍니다.

Example 30 : JSON 인코딩된 DID 문서 메타 데이터 예

```
{
  "created": "2019-03-23T06:35:22Z",
  "updated": "2023-08-10T13:40:06Z"
}
```

이 예는 다음 형식의 메타 데이터 구조에 해당합니다.

Example 29 : DID 문서 메타 데이터 예

```
«[
  "created" → "2019-03-23T06:35:22Z",
  "updated" → "2023-08-10T13:40:06Z"
]»
```

9) 보안 고려 사항

이 섹션은 비규범적입니다.

참고 : 구현자 참고 사항

Working Draft 단계에서 이 섹션은 초기 구현에서 중요시해야 하는 보안 주제에 중점을 둡니다. 편집자는 이 섹션 또는 사양의 다른 부분에 반영되어야 하는 위협 및 위협 완화에 대한 피드백을 찾고 있습니다. DID는 많은 IETF 표준에서 사용하는 일반적인 인터넷 위협 모델에서 작동하도록 설계되었습니다. 우리는 손상되지 않은 엔드 포인트를 가정하지만 네트워크에서 메시지가 읽히거나 손상시킬 수 있습니다.

(1) DID 분해자(resolver) 선택

DID 메소드 레지스트리([DID-SPEC-REGISTRIES] 참조)는 DID 메소드 이름과 해당 DID 메소드 사양의 정보 목록입니다. 개발자는 어떤 DID 메소드 사양을 사용해야 하는지에 대한 중앙 권한이 없다는 점을 명심해야 합니다. 특정 DID 메소드 이름을 사용하지만 사용할 DID 분해자 구현을 선택할 때 정보에 입각한 결정을 내리기 위해 DID 메소드 레지스트리를 사용할 수 있습니다.

(2) ID 바인딩

다음 섹션에서는 ID를 DID 및 DID 문서에 바인딩하는 방법을 설명합니다 .

① DID 및 DID 문서의 제어 증명

서명과 검증 가능한 타임 스탬프를 통해 DID 문서를 암호화 방식으로 확인할 수 있습니다. 자체적으로, 자체 서명된 DID 문서의 검증된 서명은 DID의 제어를 증명하지 않습니다. 이것은 다음만을 증명할 뿐입니다.

- 타임 스탬프가 찍힌 이후로 DID 문서가 변조되지 않았습니다.
- DID 컨트롤러는 타임 스탬프가 생성될 때 서명에 사용된 개인 키를 제어했습니다.

DID 제어를 증명하는 데는 DID와 DID 문서 간의 바인딩이 다음 두 단계 프로세스를 거쳐야 합니다.

1. 해당 DID 메소드 사양에 따라 DID 문서에 DID를 확인하는 과정
2. DID 문서가 DID와 일치한다는 결과로 나온 id 속성이 확인되었다는 검증 과정

이 프로세스는 DID 문서의 서명 여부에 관계없이 DID 및 DID 문서의 제어를 증명한다는 점에 유의해야 합니다.

DID 문서의 서명은 선택 사항입니다. DID 방법 사양은 적용 가능한 경우 구현을 설명하고 지정해야 합니다.

② 공개 키의 제어 증명

DID 문서의 공개 키 설명에 해당하는 개인 키의 제어를 증명하는 방법에는 정적 및 동적 두 가지 방법이 있습니다.

정적 방법은 개인 키로 DID 문서에 서명하는 것입니다. 이것은 늦어도 DID 문서가 등록되기 전에 개인 키의 제어를 증명합니다. DID 문서가 서명되지 않은 경우, DID 문서에 설명된 공개 키의 제어는 동적으로 다음과 같이 입증할 수 있습니다:

1. DID 문서 내의 공개 키 설명을 포함한 챌린지 메시지를 DID 문서에 설명된 적절한 서비스 엔드 포인드에 보내기
2. 공개 키 설명에 대한 응답 메시지의 서명을 검증하기

③ 인증 및 검증 가능한 클레임

DID 및 DID 문서는 본질적으로 PII(개인 식별 정보)를 수행하지는 않습니다. 예를 들어 DID와 동일한 객체를 가진 자격 증명을 사용하는 개인이나 회사와 같은 실제 세계의 어떤 것에 DID를 바인딩하는 프로세스는 이 사양의 범위를 벗어납니다. 자세한 내용은 [VC-DATA-MODEL]을 참조하십시오.

(3) 인증 서비스 엔드 포인트

DID 문서가 DID 주체 허가 및 인증을 위한 서비스 엔드 포인트에 발행되는 경우(§ 5.5 서비스 엔드 포인트 참조), 해당 서비스 엔드 포인트에서 지원되는 인증 프로토콜의 요구 사항을 준수하는 것은 서비스 엔드 포인트 공급자, 객체 또는 요청 당사자의 책임입니다.

(4) 부인 방지(Non-Repudiation)

다음과 같은 가정하에 DID 및 DID 문서 업데이트의 부인 방지가 지원됩니다.

- 무단 업데이트를 모니터링 합니다(§ 9.5 DID 문서 변경 알림 참조).
- DID 메소드에 대한 액세스 제어 메커니즘에 따라 악성 업데이트를 되돌릴 수 있는 적절한 기회가 있습니다(§ 5.4.1 인증 참조).).

타임 스탬프가 포함된 경우 부인 방지가 추가로 지원되고(§ 8.1.3 DID 문서 메타 데이터 속성 참조), 목표 DLT 시스템은 타임 스탬프를 지원합니다.

(5) DID 문서 변경 알림

DID 문서에 대한 무단 변경에 대한 한 가지 완화 방법은 변경 사항이 있을 때 DID 주체를 모니터링하고 적극적으로 알리는 것입니다. 이는 등록된 이메일 주소로 비밀번호 재설정 알림을 보내 기존 사용자 이름/비밀번호 계정에 대한 계정 탈취를 방지하는 것과 유사합니다.

DID의 경우 이러한 알림을 생성할 중개 등록 기관 또는 계정 제공 업체가 없습니다. 그러나 변경 알림을 직접적인 지원하도록 등록한 DID에 검증가능한 데이터 저장소가 있는 경우, 구독서비스가 DID 컨트롤러에 제공될 수 있습니다. 알림은 기존 DID에 나열된 관련 서비스 엔드 포인트로 직접 전송될 수 있습니다.

(검증 가능한 데이트 저장소보다) 제3자 당사자 모니터링 서비스에 의존하도록 DID 컨트롤러를 선택하는 경우, 다른 벡터 공격을 소개합니다.

(6) 키 및 서명 만료

탈중앙 ID 아키텍처에, 키 또는 서명 만료 정책을 시행하는 중앙집권 기관이 없습니다. 따라서 DID 해석자 및 기타 클라이언트 응용 프로그램은 키가 사용된 시점에 만료되지 않았는지 확인해야 합니다. 일부 사용 사례에는 이미 만료된 키를 확장할 수 있는 합법적인 이유가 있을 수 있으므로 키 만료로 인해 키가 더 이상 사용되지 않는지 확인하고 해석자의 구현이 이러한 확장 동작과 호환되어야 합니다.

(7) 키 해지 및 복구

§ 7.2 메소드 작업(operation)에는 DID 문서를 업데이트된 DID 문서로 대체하여 비활성화하는 것을 포함하여 DID 메소드 사양에서 지원할 DID 작업을 지정합니다. 또한 암호화 키 폐기가 발생하는 방법을 정의하는 것은 DID 메소드에 달려 있습니다. 또한 DID 메소드 사양은 신뢰할 수 있는 당사자의 쿼럼을 지원하여 키 복구를 가능하게 할 것으로 예상됩니다. 이를 위한 일부 기능은 § . 5.2 제어에 제안되어 있습니다. 모든 DID 메소드 사양이 다른 DID 메소드를 사용하여 등록된 DID 제어를 인식하는 것은 아니며 제3자가 동일한 방법을 사용하는 DID를 제어하는 것은 제한 할 수 있습니다. DID 메소드 사양의 액세스 제어 및 키 복구에는 복구를 위한 두 번째 제어 트랙을 유지하여 키 손상으로부터 보호하는 시간 잠금 기능도 포함될 수 있습니다. 이러한 유형의 제어에 대한 추가 사양은 향후 작업해야 할 문제입니다.

(8) 인간 친화적 식별자의 역할

DID는 중앙 등록 기관 없이도 글로벌 고유성을 달성합니다. 그러나 이것은 인간의 기억력을 비용으로 합니다. 글로벌 고유한 식별자를 생성할 수 있는 알고리즘은 인간 의미가 없는 임의의 문자열을 자동으로 생성합니다. 이것은 "인간적 의미, 탈중앙화, 보안-둘 중 하나 선택"이라는 Zooko의 Triangle에 설명된 식별자에 대한 공리를 보여줍니다 .

물론 인간에게 친숙한 식별자로 시작할 때 DID를 발견하는 것이 바람직한 사용 사례가 많이 있습니다. 예를 들어 자연어 이름, 도메인 이름 또는 DID 컨트롤러의 일반적인 주소(예 : 휴대폰 번호, 이메일 주소, Twitter 핸들 또는 블로그 URL)가 있습니다. 그러나 사람에게 친숙한 식별자를 DID에 매핑하는 문제(확인 및 신뢰할 수 있는 방식으로 수행)는 이 사양의 범위를 벗어납니다.

이 문제에 대한 해결책은 이 사양을 참조하는 별도의 사양으로 정의되어야 합니다. 이러한 사양은 다음 사항을 신중하게 고려하는 것이 좋습니다.

- 목적 엔티티에 대한 진정한 인간 친화적 식별자에 대해 사용자를 속이는 것에 기반한 수많은 보안 공격.
- 특히 글로벌적으로 고유한 경우, 본질적으로 연관성이 있는 인간 친화적인 식별자를 사용하는 경우 개인정보보호 중요도

참고

DNS 조회를 사용하여 도메인 이름 및 이메일 주소에서 DID를 검색하기 위한 초안 사양은 [DNS-DID]에서 확인할 수 있습니다.

(9) 불변성(Immutaqbility)

많은 사이버 보안 남용은 합리적이고 성실한 행위자의 가정과 현실간의 격차를 악용하는 데 달려 있습니다. 다른 생태계와 마찬가지로 DID 생태계에도 이것이 발생할 가능성이 있습니다. 이 사양은 프로토콜 대신 데이터 모델에 초점을 맞추기 때문에 해당 모델이 어떻게 사용되는지에 대한 많은 측면에 대한 의견을 제공하지 않습니다. 그러나 개별 DID 방법은 필요하지 않은 동작이나 의미를 제거하는 제약 조건을 고려할 수 있습니다. 동일 특색을 제공하면서 더 많은 locked down DID 방법이 악의적인 행위자에 의해 덜 처리될 수 있습니다.

예를 들어 데이터 모델이 업데이트와 관련하여 제공하는 유연성을 고려하십시오. DID 문서에 대한 단일 편집은 해당 문서의 루트 id속성을 제외한 모든 것을 변경할 수 있습니다. 데이터 모델의 개별 JSON 개체는 id를 제외한 모든 속성들을 변화시킬 수 있습니다. 그러나 서비스 엔드 포인트에서 정의된 후에 type을 변경 하는 것이 실제로 바람직합니까? 아니면 키의 값을 변경하려면? 아니면 객체의 특정 기본 속성이 변경되면 새로운 id을 요구하는 것이 더 나을까요? 악의적인 웹 사이트 탈취는 종종 사이트가 식별자(호스트 이름)를 유지하지만 그 아래에 미묘하고 위험한 변경 사항이 있는 결과를 목표로 합니다. 사양에서 사이트의 특정 속성이 변경 불가능하도록 요구하는 경우(예 : IP 주소와 관련된 ASN) 이러한 공격은 수행하기가 훨씬 더 어렵고 비용이 많이 들 수 있으며 이상 탐지가 더 쉬울 것입니다.

불변성이 일부 사이버 보안 이점을 제공한다는 개념은 캐싱 때문에 특히 관련이 있습니다. 신뢰할 수 있는 글로벌 소스와 관련된 DID 문서의 경우, DID 문서 최신 버전의 적시(just-in-time), 직접적 조회가 가능합니다. 그러나 캐시 레이어는 결국 클라이언트와 해당 소스 사이에 위치할 수 있습니다. 그렇게 한다면 실제로 미묘하게 다르지만 DID 문서에 있는 개체의 속성이 주어진 상태를 가지고 있다고 믿으면 악용을 유발할 수 있습니다. 이는 일부 조회가 전체 DID 문서이고 다른 조회가 더 큰 컨텍스트를 가정하는 부분 데이터인 경우 특히 그렇습니다.

(10) DID 문서의 암호화된 데이터

DID 문서는 일반적으로 공개적으로 사용할 수 있습니다. 암호화 알고리즘은 암호화 및 컴퓨팅 성능의 발전으로 인해 실패하는 것으로 알려져 있습니다. 개발자는 DID 문서에 있는 암호화된 데이터가 결국 암호화된 데이터를 사용할 수 있는 동일한 대상에게 일반 텍스트로 제공될 수 있다고 가정하는 것이 좋습니다.

DID 문서의 전체 또는 일부를 암호화하는 것은 장기적으로 데이터를 보호하는 적절한 수단이 아닙니다. 마찬가지로 DID 문서에 암호화된 데이터를 배치하는 것은 개인 식별 정보를 포함하는

적절한 수단이 아닙니다.

위의 주의사항을 감안할 때 암호화된 데이터가 DID 문서에 포함된 경우 구현자는 DID와의 상관관계를 원하지 않는 엔티티의 공개 키로 암호화하지 않는 것이 좋습니다.

10) 개인정보 고려 사항

이 섹션은 비규범적입니다.

DID 및 DID 문서는 설계상 DID 컨트롤러가 직접 관리하기 때문에 개인정보보호 원칙을 탈중앙 ID 아키텍처의 모든 측면에 적용하는 것이 매우 중요합니다. 추가 개인정보보호 장치를 권장하거나 적용할 등록 기관, 호스팅 회사 또는 기타 중간 서비스 제공 업체가 없습니다. 이 사양의 작성자는 개발 전반에 걸쳐 7가지 Privacy by Design 원칙을 모두 적용해야 합니다. 예를 들어, 이 사양의 개인정보보호는 교정이 아닌 예방적이며 개인정보보호는 내장된 기본값입니다. 또한 탈 중앙화된 식별자 아키텍처는 그 자체로 "사용자 프라이버시 존중-사용자 중심 유지"라는 원칙 # 7을 구현합니다.

이 섹션에는 구현자, 위임자 및 DID 객체가 염두에 두어야 하는 추가 개인정보 고려 사항이 나열되어 있습니다.

(1) 개인 식별 정보(PII)를 비공개로 유지

DID 메소드 사양이 공공 검증 데이터 레지스트리로 작성된 경우, 모든 DID와 DID 문서가 공개되어야 하며, DID 문서에 개인정보를 포함하지 않는 것이 중요합니다. 모든 개인 데이터는 DID 주체의 통제하에 서비스 엔드 포인트 뒤에 보관되어야 합니다. 서비스 엔드 포인트의 URL 내에서 의도하지 않은 개인 데이터 또는 상관관계의 유출을 방지하기 위해 서비스 엔드 포인트에서 URL 사용에 대한 추가 검수를 수행해야 합니다. 예를 들어 사용자 이름이 포함된 URL은 DID 문서에 포함하기에 위험할 수 있습니다. 사용자 이름은 DID 객체가 공유에 동의하지 않은 정보를 의도하지 않게 공개할 수 있는 개인적인 의미를 가질 수 있기 때문입니다. 이 개인정보 아키텍처를 통해 개인 데이터는 DID 문서의 공개 키 설명에 의해 식별되고 보호되는 통신 채널을 사용하여 개인 P2P 기반으로 교환될 수 있습니다. 이것은 또한 개인 데이터가 불변의 분산 원장에 기록되지 않기 때문에 DID 객체와 요청 당사자가 잊혀질 GDPR 권리를 구현할 수 있도록 합니다.

(2) DID 상관관계 위험 및 가명 DID

모든 유형의 글로벌 고유 식별자와 마찬가지로 DID는 상관관계에 사용될 수 있습니다. DID 컨트롤러는 모든 관계에 대해 서로 다른 개인 DID를 공유하는 쌍별 고유 DID를 사용함으로, 이러한 개인정보 위험을 완화할 수 있습니다. 실제로 각 DID는 가명 역할을 합니다. DID 객체가 해당 당사자 간의 상관관계를 명시적으로 승인하는 경우, 가명 DID는 둘 이상의 당사자와만 공유됩니다. 익명 DID가 기본값인 경우 공개 DID(공개적으로 게시되거나 많은 당사자와 공유되는 DID)는 DID 객체가 공개적으로 신원을 밝히기는 것만 필요로 합니다.

(3) DID 문서 상관관계 위험

DID 문서와 관련한 데이터가 상관관계가 될 수 있는 경우, 가명 DID의 상관관계 방지 보호는 쉽게 무효화됩니다. 예를 들어, 여러 DID 문서에서 동일한 공개 키 설명 또는 맞춤형 서비스 엔드 포인트를 사용하면 동일한 DID를 사용하는 것 만큼 많은 상관 정보를 제공할 수 있습니다. 따라서 가명 DID에 대한 DID 문서도 쌍으로 고유한 공개 키를 사용해야 합니다. 익명의 DID에 대해 DID 문서에서 쌍으로 고유한 서비스 엔드 포인트를 사용하는 것도 자연스러워 보일 수 있습니다.. 그러나 고유한 엔드 포인트를 사용하면 두 DID 간의 모든 트래픽을 타이밍 상관관계 및 유사한 분석이 쉬운 고유한 버킷으로 완벽하게 격리할 수 있습니다. 따라서 엔드 포인트 개인정보보호를 위한 더 나은 전략은 많은 다른 객체들이 제어하는 수천 또는 수백만 개의 DID 간에 엔드 포인트를 공유하는 것입니다.

(4) Herd 개인정보보호

DID 객체가 herd에서 다른 사람과 구별되는 경우, 개인정보보호가 가능할 수 있습니다. 다른 당사자와 개인적으로 관여하는 행위가 그 자체로 인식 할 수 있는 플래그일 때 개인정보보호는 크게 약화됩니다. DID 및 DID 방법은 특히 합법적으로 가장 필요로 하는 사람들을 위해 herd 개인정보보호를 개선하기 위해 노력해야 합니다. 기본적으로 익명성과 가명성을 유지하는 기술과 휴먼 인터페이스를 선택하십시오. 디지털 지문을 줄이려면 클라이언트 구현 간에 공통 설정을 공유하고, 유선 프로토콜에서 협상된 옵션을 최소화하고, 암호화된 전송 레이어를 사용하고, 메시지를 표준 길이로 채우십시오.

11) 예제

이 섹션은 비규범적입니다.

(1) DID 문서

이 섹션은 비규범적입니다.

선택적 확장 및 기타 검증 방법 유형은 did-spec-registries를 참조하십시오 .

참고

이러한 예는 정보 제공 목적으로만 사용되며 여러 목적으로 동일한 검증 방법을 사용하지 않는 것이 좋습니다.

Example 32 : 1 가지 검증 방법 유형의 DID 문서

```
{
  "@context": "https://www.w3.org/ns/did/v1",
  "id": "did:example:123",
  "authentication": [
    {
      "id": "did:example:123#z6MkecaLyHuYWkayBDLw5ihndj3T1m6zKTGqau3A51G7RBf3",
      "type": "Ed25519VerificationKey2018",
      "controller": "did:example:123",
      "publicKeyBase58": "AKJP3f7BD6W4iWEQ9jwndVTCBq8ua2Utt8EEjJ6Vxsf"
    }
  ],
  "capabilityInvocation": [
    {
      "id": "did:example:123#z6MkhdmzFu659ZJ4XKj31vtEDmjvsi5yDZG5L7Caz63oP39k",
      "type": "Ed25519VerificationKey2018",
      "controller": "did:example:123",
      "publicKeyBase58": "4BWwfeqdp1obQptLLMvPNgBw48p7og1ie6Hf9p5nTpNN"
```

```
        }
    ],
    "capabilityDelegation": [
        {
            "id": "did:example:123#z6Mkw94ByR26zMSkNdCUi6FNRsWnc2DFEeDXyBGJ5KT
zSWyi",
            "type": "Ed25519VerificationKey2018",
            "controller": "did:example:123",
            "publicKeyBase58": "Hgo9PAmfeoxHG8Mn2XHXamxnnSwPpkyBHAMNF3VyXJCL"
        }
    ],
    "assertionMethod": [
        {
            "id":   "did:example:123#z6MkiukuAuQAE8ozxvmahnQGzApvtW7KT5XXKfojjwb
dEomY",
            "type": "Ed25519VerificationKey2018",
            "controller": "did:example:123",
            "publicKeyBase58": "5TVraf9itbKXrRvt2DSS95Gw4vqU3CHAdetoufdcKazA"
        }
    ]
}
```

Example 33 : 다양한 검증 방법을 가진 DID 문서


```
{
  "@context": "https://www.w3.org/ns/did/v1",
  "id": "did:example:123",
  "verificationMethod": [
    {
      "id": "did:example:123#ZC2jXTO6t4R501bfCXv3RxarZyUbdP2w_psLwMuY6ec",
      "type": "Ed25519VerificationKey2018",
      "controller": "did:example:123",
      "publicKeyBase58": "H3C2AVvLMv6gmMNam3uVAjZpfkcJCwDwnZn6z3wXmqPV"
    },
    {
      "id": "did:example:123#zQ3shP2mWsZYWgvgM11nenXRTx9L1yiJKmkf9dfX7NaMKb1pX",
      "type": "EcdsaSecp256k1VerificationKey2019",
      "controller": "did:example:123",
      "publicKeyBase58": "d5cW2R53NHTTkv7EQSYR8YxaKx7MVCcchjmK5EgCNXxo",
    },
    {
      "id": "did:example:123#_Qq0UL2Fq651Q0Fjd6TvnYE-faHiOpRlPVQcY_-tA4A",
      "type": "JsonWebKey2020",
      "controller": "did:example:123",
      "publicKeyJwk": {
        "kty": "OKP",
        "crv": "Ed25519",
        "x": "VCpo2LMLhn6iWku8MKvSLg2ZAoC-nlOyPVQaO3FxVeQ"
      }
    },
    {
      "id": "did:example:123#z6LSnjagzhe8Df6gZmroW3wjDd7XQLwAuYfwa4ZeTBCGFoYc",
      "type": "JsonWebKey2020",
```

```
      "controller": "did:example:123",
      "publicKeyJwk": {
        "kty": "OKP",
        "crv": "X25519",
        "x": "pE_mG098rdQjY3MKK2D5SUQ6ZOEW3a6Z6T7Z4SgnzCE"
      },
    }
    {
      "id": "did:example:123#4SZ-StXrp5Yd4_4rxHVTCYTHyt4zyPfN1fIuYsm6k3A",
      "type": "JsonWebKey2020",
      "controller": "did:example:123",
      "publicKeyJwk": {
        "kty": "EC",
        "crv": "secp256k1",
        "x": "Z4Y3NNOxv0J6tCgqOBFnHnaZhJF6LdulT7z8A-2D5_8",
        "y": "i5a2NtJoUKXkLm6q8nOEu9WOkso1Ag6FTUT6k_LMnGk"
      }
    },
    {
      "id": "did:example:123#n4cQ-I_WkHMcwXBJa7IHkYu8CMfdNcZKnKsOrnHLpFs",
      "type": "JsonWebKey2020",
      "controller": "did:example:123",
      "publicKeyJwk": {
        "kty": "RSA",
        "e": "AQAB",
       "n":
"omwsC1AqEk6whvxyOltCFWheSQvv1MExu5RLCMT4jVk9khJKv8JeMXWe3bWHatjPskdf2
dlaGkW5QjtOnUKL742mvr4tCldKS3ULIaT1hJInMHHxj2gcubO6eEegACQ4QSu9LO0H-L
M_L3DsRABB7Qja8HecpyuspW1Tu_DbqxcSnwendamwL52V17eKhlO4uXwv2HFlxufFHM
0KmCJujIKyAxjD_m3q__IiHUVHD1tDIEvLPhG9Azsn3j95d-saIgZzPLhQFiKluGvsjrSkYU5p
```

```
XVWIsV-B2jtLeeLC14XcYxWDUJ0qVopxkBvdlERcNtgF4dvW4X00EHj4vCljFw"
    },
    {
      "id": "did:example:123#_TKzHv2jFIyvdTGF1Dsgwngfdg3SH6TpDv0Ta1aOEkw",
      "type": "JsonWebKey2020",
      "controller": "did:example:123",
      "publicKeyJwk": {
        "kty": "EC"
        "crv": "P-256",
        "x": "38M1FDts7Oea7urmseiugGW7tWc3mLpJh6rKe7xINZ8",
        "y": "nDQW6XZ7b_u2Sy9slofYLlG03sOEoug3I0aAPQ0exs4"
      }
    },
    {
      "id": "did:example:123#8wgRfY3sWmzoeAL-78-oALNvNj67ZlQxd1ss_NX1hZY",
      "type": "JsonWebKey2020",
      "controller": "did:example:123",
      "publicKeyJwk": {
        "kty": "EC",
        "crv": "P-384",
        "x": "GnLl6mDti7a2VUIZP5w6pcRX8q5nvEIgB3Q_5RI2p9F_QVsaAlDN7IG68Jn0dS_F",
        "y": "jq4QoAHKiIzezDp88s_cxSPXtuXYFliuCGndgU4Qp8l91xzD1spCmFIzQgVjqvcP"
      }
    },
    {
      "id": "did:example:123#NjQ6Y_ZMj6IUK_XkgCDwtKHlNTUTVjEYOWZtxhp1n-E",
      "type": "JsonWebKey2020",
      "controller": "did:example:123",
      "publicKeyJwk": {
        "kty": "EC",
```

```
            "crv": "P-521",
            "x": "AVlZG23LyXYwlbjbGPMxZbHmJpDSu-IvpuKigEN2pzgWtSo--Rwd-n78nrWn
ZzeDc187Ln3qHlw5LRGrX4qgLQ-y",
            "y": "ANIbFeRdPHf1WYMCUjcPz-ZhecZFybOqLIJjVOlLETH7uPlyG0gEoMWnIZXh
QVypPy_HtUiUzdnSEPAylYhHBTX2"
        }
    }
  ]
}
```


(2) 증명(Proving)

이 섹션은 비규범적입니다.

참고

이러한 예제는 정보 제공 목적으로만 사용됩니다. 추가 예제는 W3C 검증 가능한 자격 증명 데이터 모델을 참조하십시오.

Example 34 : Ed25519VerificationKey2018 유형의 검증 방법에 연결된 검증 가능한 자격 증명


```
{
  "@context": [
    "https://www.w3.org/2018/credentials/v1",
    "https://w3id.org/citizenship/v1"
  ],
  "type": [
    "VerifiableCredential",
    "PermanentResidentCard"
  ],
  "credentialSubject": {
    "id": "did:example:123",
    "type": [
      "PermanentResident",
      "Person"
    ],
    "givenName": "JOHN",
    "familyName": "SMITH",
    "gender": "Male",
    "image": "...kJggg==",
    "residentSince": "2015-01-01",
    "lprCategory": "C09",
    "lprNumber": "000-000-204",
    "commuterClassification": "C1",
    "birthCountry": "Bahamas",
    "birthDate": "1958-08-17"
  },
  "issuer": "did:example:456",
  "issuanceDate": "2020-04-22T10:37:22Z",
  "identifier": "83627465",
```

```
  "name": "Permanent Resident Card",
  "description": "Government of Example Permanent Resident Card.",
  "proof": {
    "type": "Ed25519Signature2018",
    "created": "2020-04-22T10:37:22Z",
    "proofPurpose": "assertionMethod",
    "verificationMethod": "did:example:456#key-1",
    "jws": "eyJjcml0IjpbImI2NCJdLCJiNjQiOmZhbHNlLCJhbGciOiJFZERTQSJ9..BhWew0x-txcroGjgdtK-yBCqoetg9DD9SgV4245TmXJi-PmqFzux6Cwaph0r-mbqzlE17yLebjfqbRT275U1AA"
  }
}
```

Example 35 : JsonWebKey2020 유형의 검증 방법에 연결된 검증 가능한 자격 증명


```
{
  "@context": [
    "https://www.w3.org/2018/credentials/v1",
    "https://www.w3.org/2018/credentials/examples/v1"
  ],
  "id": "http://example.gov/credentials/3732",
  "type": ["VerifiableCredential", "UniversityDegreeCredential"],
  "issuer": { "id": "did:example:123" },
  "issuanceDate": "2020-03-10T04:24:12.164Z",
  "credentialSubject": {
    "id": "did:example:456",
    "degree": {
      "type": "BachelorDegree",
      "name": "Bachelor of Science and Arts"
    }
  },
  "proof": {
    "type": "JsonWebSignature2020",
    "created": "2020-02-15T17:13:18Z",
    "verificationMethod": "did:example:123#_Qq0UL2Fq651Q0Fjd6TvnYE-faHiOpRlPVQcY_-tA4A",
    "proofPurpose": "assertionMethod",
    "jws": "eyJiNjQiOmZhbHNlLCJjcml0IjpbImI2NCJdLCJhbGciOiJFZERTQSJ9..Y0KqovWCPAee
FhkJxfQ22pbVl43Z7UI-X-1JX32CA9MkFHkmNprcNj9Da4Q4QOl0cY3obF8cdDRdnKr0IwNrAw"
  }
}
```

Example 36 : 디코딩된 JWT 로 검증 가능한 자격 증명


```
{
  "protected": {
    "kid": "did:example:123#_Qq0UL2Fq651Q0Fjd6TvnYE-faHiOpRlPVQcY_-tA4A",
    "alg": "EdDSA"
  },
  "payload": {
    "iss": "did:example:123",
    "sub": "did:example:456",
    "vc": {
      "@context": [
        "https://www.w3.org/2018/credentials/v1",
        "https://www.w3.org/2018/credentials/examples/v1"
      ],
      "id": "http://example.gov/credentials/3732",
      "type": [
        "VerifiableCredential",
        "UniversityDegreeCredential"
      ],
      "issuer": {
        "id": "did:example:123"
      },
      "issuanceDate": "2020-03-10T04:24:12.164Z",
      "credentialSubject": {
        "id": "did:example:456",
        "degree": {
          "type": "BachelorDegree",
          "name": "Bachelor of Science and Arts"
        }
      }
```

```
    },
    "jti": "http://example.gov/credentials/3732",
    "nbf": 1583814252
  },
  "signature": "qSv6dpZJGFybtcifLwGf4ujzlEu-fam_M7HPxinCbVhz9iIJCg70UMeQbPa1ex
6BmQ2tnSS7F11FHnMB2bJRAw"
}
```

(3) 암호화(Encrypting)

이 섹션은 비규범적입니다.

참고

이러한 예는 정보 제공 목적으로만 사용되며 JWE 헤더에서 불필요한 정보가 공개되지 않도록 하는 것이 모범 사례로 간주됩니다.

Example 37 : kid 를 통해 검증 방법에 연결된 JWE

```
{
  "ciphertext": "3SHQQJajNH6q0fyAHmw...",
  "iv": "QldSPLVnFf2-VXcNLza6mbylYwphW57Q",
  "protected": "eyJlbmMiOiJYQzIwUCJ9",
  "recipients": [
    {
      "encrypted_key": "BMJ19zK12YHftJ4sr6Pz1rX1HtYni_L9DZvO1cEZfRWDN2vXeOYlwA",
      "header": {
        "alg": "ECDH-ES+A256KW",
        "apu": "Tx9qG69ZfodhRos-8qfhTPc6ZFnNUcgNDVdHqX1UR3s",
        "apv": "ZGlkOmVsZW06cm9wc3RlbjpFa...",
        "epk": {
          "crv": "X25519",
          "kty": "OKP",
          "x": "Tx9qG69ZfodhRos-8qfhTPc6ZFnNUcgNDVdHqX1UR3s"
        },
        "kid": "did:example:123#zC1Rnuvw9rVa6E5TKF4uQVRuQuaCpVgB81Um2u17Fu7UK"
      }
    }
  ],
  "tag": "xbfwwDkzOAJfSVem0jr1bA"
}
```

A. 현재 이슈

아래 이슈 목록은 활발히 논의 중이며 이 사양이 변경될 가능성이 있습니다.

확장성 마감 대기 중

이슈 5 : DID 컨텍스트는 어디에 있습니까?

DID 컨텍스트는 어디에 있습니까?

jose keys need-special-call 마감 대기 중

이슈 8 : 암호화 알고리즘 지정하는 레지터리 RFC7528

암호화 알고리즘 지정하는 레지터리 RFC7518

마감 대기 중

이슈 9 : RsaSignature2017을 표준 JWA 서명으로 교체

RsaSignature2017을 표준 JWA 서명으로 대체

마감 대기 중

이슈 10 : RsaSignature2018 설명

RsaSignature2018 설명

마감 대기 중

이슈 14 : 키 해지 목록 표준화

키 해지 목록 표준화

논의 중 확장성 높은 우선 순위

이슈 23 : 컨텍스트에서 누락된 publicKeyJwk, publicKeyHex, publicKeyBase64, publicKeyBase58이

컨텍스트에서 publicKeyJwk, publicKeyHex, publicKeyBase64, publicKeyBase58이 누락되었습니다.

논의 중 need-special-call

이슈 33 : 표준 DID을 사용한 ledgers간 DID를 이동하는 옵션과 저렴한 DIDs

표준 DID을 사용한 ledgers 간 DID를 이동하는 옵션과 저렴한 DIDs

PR 존재

이슈 36 : 방법별 DID 매개 변수 사용에 대한 세부 사항

방법별 DID 매개 변수 사용에 대한 세부 사항

논의 중 높은 우선 순위

이슈 55 : @context에서 ethereumAddress 공개 키 유형에 대한 지원 추가

@context에 ethereumAddress 공개 키 유형에 대한 지원 추가

논의 중 마감 대기 중

이슈 57 : 인증 섹션의 다른 검증 방법에 누락에 대한 해명

인증 섹션에 다른 검증 방법이 누락된 것에 대한 해염

논의 중 확장성

이슈 58 : 레지스트리 처리

레지스트리 처리

논의 중 메타 데이터

이슈 65 : DID 문서 메타 데이터가 문서에 속합니까?

DID 문서 메타 데이터가 문서에 속합니까?

논의 중 편집 메타 데이터

이슈 72 : 개인정보 보호 고려 사항-특히 GDPR 호출

개인정보 고려 사항-특히 GDPR을 호출

마감 대기 중

이슈 75 : 공개 키 폐기 추적

공개 키 폐기 추적

논의 중 높은 우선 순위 메타 데이터

이슈 85 : 응용 데이터에 비해 DID에 관해 구문적으로 구분되는 데이터

응용 데이터에 비해 DID에 관해 구문적으로 구분되는 데이터

PR 존재 논의 중

이슈 92 : JSON-LD과 동등하고 JSON과 유사한 유효한 유형의 DID 문서 구문에 CBOR 추가

JSON-LD과 동등하고 JSON과 유사한 유효한 유형의 DID 문서 구문에 CBOR 추가

논의 중 수평적 검토

이슈 94 : DID 설명자 생성

DID 설명자 생성

논의중

이슈 95: 문서 구조

문서 구조

수평적 검토 i18n- 추적기

이슈 104 : 수평적 검토 : 자체 테스트 국제화

수평적 검토 : 자체 테스트 국제화

a11y- 트래커 수평적 검토

이슈 105 : 수평적 검토 : 접근성 자체 테스트

수평적 검토 : 접근성 자체 테스트

편집 CR 직전

이슈 118 : 사양이 WCAG 2.0을 준수해야 함

사양은 WCAG 2.0을 준수해야 합니다.

수평적 검토

이슈 119 : 수평 적 검토 : TAG에 검토 기회 제공

수평적 검토 : TAG에 검토 기회 제공

PR 존재

이슈 122 : DID 객체가 DID 컨트롤러가 아닌 경우는 언제입니까?

DID 객체가 DID 컨트롤러가 아닌 경우는 언제입니까?

PR 존재

이슈 137 : DID 매개 변수가 사양에서 규범적이어야 합니까?

사양에서 DID 매개 변수가 규범 적이어야 합니까?

편집

이슈 151 : eIDAS 보증 수준에 대한 논의 포함

eIDAS 보증 수준에 대한 논의 포함

확장성 마감 대기 중

이슈 154 : LD-Proof / LDS 사양에서 DID 코어 사양 분리

LD-Proof / LDS 사양에서 DID Core 사양 분리

편집 CR 직전

이슈 163 : 사양에 정의된 용어의 사용은 정의에 연결되어 있어야 함.

사양에 정의된 용어의 사용은 해당 정의에 대해 연결되어 있어야 함.

편집 마감 대기 중

이슈 165 : 엔티티쉽 및 권한 시작(SOA) 문제는 무엇입니까?

엔티티쉽 및 권한 시작(SOA) 문제는 무엇입니까?

확장성

이슈 169 : 커뮤니티 그룹에서 관리하는 레지스트리를 이 사양에서 설정한 레지스트리로 교체

커뮤니티 그룹이 관리하는 레지스트리를 이 사양에 의해 설정된 레지스트리로 대체합니다.

편집 jose need-special-call

이슈 170 : 공개 키 "id" 및 "type" 멤버가 JWK "kid" 및 "kty" 멤버를 복제함

공개 키 "id" 및 "type" 멤버는 JWK "kid" 및 "kty" 멤버를 복제합니다.

편집 jose need-special-call

이슈 171 : JWK를 사용하여 공개 키 예제 추가

JWK를 사용하여 공개 키 예제 추가

편집 메타 데이터

이슈 174 : "업데이트된" 속성의 지정되지 않은 의미

"업데이트된" 속성의 지정되지 않은 의미

편집

이슈 176 : 침해 시 공격으로부터 보호에 대한 입증되지 않은 진술

침해 시 공격으로부터 보호에 대한 입증되지 않은 진술

편집 마감 대기 중

이슈 178 : 타임 스탬프와 서명을 결합하는 것에 대한 명시되지 않은 진술

타임 스탬프와 서명을 결합하는 것에 대한 명시되지 않은 진술

jose

이슈 185 : DID 문서에서 지원되는 암호

DID 문서에서 지원되는 암호

논의 중 마감 대기 중

이슈 190 : 이슈 4에서 논의되는 내용(사용 사례를 통한 TERMX 설명, 사양 포인터 및 PR)

이슈 4에서 논의된 내용(사용 사례를 통한 TERMX 설명, 사양 포인터 및 PR)

논의 중

이슈 195 : DID 문서 작업에 대해 승인된 검증 방법이 명확하지 않음

DID 문서 작업에 대해 승인된 검증 방법이 명확하지 않음

대기 중

이슈 198 : DID 확인에 섹션 추가 마감

DID 확인에 섹션 추가

논의 중

이슈 199 : DID가 식별할 수 있는 항목에 대한 설명

DID가 식별할 수 있는 항목에 대한 설명

PR 존재 확장성

이슈 202 : 레지스트리의 JSON-LD 컨텍스트

레지스트리의 JSON-LD 컨텍스트

PR 존재 메타 데이터

이슈 203 : DID 문서 메타 데이터 정의

DID 문서 메타 데이터 정의

편집

이슈 204 : 속성 및 값에 대한 용어 정의 PR 존재

속성 및 값에 대한 용어 정의

논의 중 확장성

이슈 205 : 알 수 없는 속성을 처리하는 방법

알려지지 않은 속성을 처리하는 방법

확장성 마감 대기 중

이슈 207 : 확장성 및 적합성에 대한 섹션 추가 편집

확장성 및 적합성에 대한 섹션 추가

확장성

이슈 208 : IETF did + ld + json 미디어 유형 등록

IETF did + ld + json 미디어 유형 등록

마감 대기 중

이슈 236 : 현재 사양에서 사용되지 않는 publicKeyHex 형식

현재 사양에서 사용되지 않는 publicKeyHex 형식

Jose needs-special-call

이슈 240 : did-core가 JWK 사용을 제한해야 합니까?

did-core가 JWK 사용을 제한해야 합니까?

이슈 248 : 신뢰 당사자에 대한 용어 필요

신뢰 당사자에 대한 용어 필요

이슈 249 : "Universal Resolver에 대한 신뢰"/wrt 실패 단일 소스를 완화하는 방법?

"Universal Resolver에 대한 신뢰"/wrt 실패 단일 소스를 완화하는 방법?

이슈 253

DID 확인 및 역참조 계약이 추가됨

마감 대기 중

이슈 258 : 사양을 준수하는 초기 구현 목록?

사양을 준수하는 초기 구현 목록?

논의 중 마감 대기 중

이슈 259 : DID 및 JOSE : publicKey.id 및 publicKey.publicKeyJwk.kid

DID 및 JOSE : publicKey.id 및 publicKey.publicKeyJwk.kid

편집 마감 대기 중 질문

이슈 260 : DID가 둘 이상의 컨트롤러를 가질 수 있는 방법에 대한 명확한 설명

DID가 두 개 이상의 컨트롤러를 가질 수 있는 방법에 대한 명확한 설명

편집

이슈 261 : SSI 원칙과 관련하여 "클라이언트"라는 용어의 정의

SSI 원칙과 관련하여 "클라이언트"라는 용어의 정의

확장성 마감 대기 중 질문

이슈 266 : DID가 자체 서명된 인증서를 지원해야 합니까?

DID가 자체 서명된 인증서를 지원해야 합니까?

이슈 267

핵심 사항을 앞에 배치

PR 존재 편집

이슈 268 : 특정 애플리케이션 레이어 사용에 대해 어느 정도의 증명 목적을 정의해야 합니까?

특정 애플리케이션 레이어 사용에 대해 어느 정도의 증명 목적을 정의해야 합니까?

PR 존재 편집 질문

이슈 269 : 통제권의 이전 및 식별자의 객체와의 교차

통제권의 이전과 식별자의 객체와의 교차

질문

이슈 270 : did 매개 변수 동등성

매개 변수 동등성

확장성 질문

이슈 272 : 사양에서 지정되지 않은 모든 속성/기능 제거

사양에서 지정되지 않은 모든 속성/기능을 제거

마감 대기 중

이슈 273 : 증명 목적과 검증 방법 간의 매핑 반전?

증명 목적과 검증 방법 간의 매핑을 반전합니까?

질문

이슈 274 : 채워진 최상위 DID 문서 'id' 속성의 필요성에 대한 모호함

채워진 최상위 DID 문서 'id' 속성의 필요성에 대한 모호함

편집 마감 대기 중

이슈 280 : publicKeyHex 사용 제거

publicKeyHex 사용 제거

편집 keys 마감 대기 중

이슈 281 : 지원되는 키 표현 publicKeyJwk, publcKeyPem 및 publicKeyBase58에 필요한 사양

지원되는 키 표현 publicKeyJwk, publcKeyPem 및 publicKeyBase58에 필요한 사양

이슈 282

CBOR 섹션 추가

PR 존재

이슈 283 : 검증 방법 블록은 공개 키와 같은 첫 번째 citizen이어야 합니다.

검증 방법 블록은 공개 키와 같은 첫 번째 citizen이어야 합니다.

질문

이슈 289 : DID 방법은 DID 작업에 대한 증명 목적을 노출해야 합니까?

DID 방법은 DID 작업에 대한 증명 목적을 노출해야 합니까?

수평적 검토

이슈 291 : PING 수평적 검토

PING 수평적 검토

수평적 검토

이슈 292 : 수평적 검토 추적

수평적 검토 추적

PR 존재

이슈 293 : 'proof' 제거

proof 제거

마감 대기 중

이슈 294 : 증명 목적을 정의하기 위해 별도의 최상위 블록 생성

증명 목적을 정의하기 위해 별도의 최상위 블록을 생성

이슈 295

유형이 없는 단순한 확인(resolution) 함수 정의

이슈 296

데이터 유형으로 확인(resolution) 함수 정의

이슈 297

데이터 유형 및 속성 값으로 확인(resolution) 함수 정의

이슈 298

데이터 유형, 속성 값 및 간단한 메타 데이터 구조로 확인(resolution) 함수 정의

이슈 299

데이터 유형, 속성 값 및 전체 메타 데이터 구조로 확인(resolution) 함수 정의

이슈 300

변환 없이 데이터 유형, 속성 값 및 전체 메타 데이터 구조로 확인(resolution) 함수 정의

B. IANA 고려 사항

이 섹션은 이 사양이 W3C 제안 권장 사항이 될 때 IANA의 검토, 승인 및 등록을 위해 IESG (Internet Engineering Steering Group)에 제출됩니다.

B.1 응용 프로그램 / did + json

Type name:

application

Subtype name :

did + json

Required parameters :

없음

Optional parameters:

없음

인코딩 고려 사항:

RFC 8259, 섹션 11을 참조하십시오.

보안 고려 사항:

RFC 8259, 섹션 12 [RFC8259]를 참조하십시오.

상호 운용성 고려 사항:

해당 없음

게시된 사양:

http://www.w3.org/TR/did-core/

이 미디어 유형을 사용하는 응용 프로그램 :

분산되고 영구적이며 암호화로 검증 가능하고 확인 가능한 식별자가 필요한 모든 애플리케이션. 애플리케이션은 일반적으로 암호화 ID 시스템, 분산형 장치 네트워크, W3C 검증 가능한 자격

증명을 발급하거나 검증하는 웹 사이트로 구성됩니다.

추가 정보:

Magic number:

해당 없음

파일 확장자 :

.did

Macintosh 파일 유형 코드 :

TEXT

추가 정보를 위해 연락할 사람 및 이메일 주소 :

Ivan Herman 〈ivan@w3.org〉

용도 :

일반

사용 제한 :

없음

저자 :

Drummond Reed, Manu Sporny, Markus Sabadello, Dave Longley, Christopher Allen

컨트롤러 변경 :

W3C

application / did + json 과 함께 사용되는 프리그먼트 식별자는 DID Core v1.0, Fragment [DID-CORE]에 정의된 규칙에 따라 처리됩니다.

B.2 응용 프로그램 / did + ld + json

유형 이름:

application

하위 유형 이름 :

did + ld + json

필수 매개 변수 :

없음

선택적 매개 변수 :

없음

인코딩 고려 사항 :

RFC 8259, 섹션 11을 참조하십시오.

보안 고려 사항 :

JSON-LD 1.1, 보안 고려 사항 [JSON-LD11] 참조하십시오.

상호 운용성 고려 사항 :

해당 없음

게시된 사양 :

http://www.w3.org/TR/did-core/

이 미디어 유형을 사용하는 응용 프로그램 :

분산되고 영구적이며 암호화로 확인 가능하고 해결 가능한 식별자가 필요한 모든 애플리케이션. 애플리케이션은 일반적으로 암호화 ID 시스템, 분산형 장치 네트워크, W3C 검증 가능한 자격 증명을 발급하거나 증명하는 웹 사이트로 구성됩니다 .

추가 정보 :

Magic number :

해당 없음

파일 확장자 :
.did

Macintosh 파일 유형 코드 :
TEXT

추가 정보를 위해 연락할 사람 및 이메일 주소 :
Ivan Herman 〈ivan@w3.org〉

용도 :
일반

사용 제한 :
없음

저자 :
Drummond Reed, Manu Sporny, Markus Sabadello, Dave Longley, Christopher Allen

컨트롤러 변경 :
W3C

application / did + ld + json 과 함께 사용되는 프리그먼트 식별자는 JSON-LD 1.1 : application / ld + json 미디어 유형 [JSON-LD11]과 관련된 규칙에 따라 처리됩니다.

B.3 응용 프로그램 / did + cbor

유형 이름 :
application

하위 유형 이름 :
did + cbor

필수 매개 변수 :
없음

선택적 매개 변수 :
없음

인코딩 고려 사항 :
RFC 7049, 섹션 4.2를 참조하십시오.

보안 고려 사항 :
RFC 7049, 섹션 10 [RFC7049]를 참조하십시오.

상호 운용성 고려 사항 :
해당 없음

게시된 사양 :
http://www.w3.org/TR/did-core/

이 미디어 유형을 사용하는 응용 프로그램 :
분산되고 영구적이며 암호화로 검증 가능하고 확인 가능한 식별자가 필요한 모든 애플리케이션. 애플리케이션은 일반적으로 암호화 ID 시스템, 분산형 장치 네트워크, W3C 검증 가능한 자격 증명을 발급하거나 확인하는 웹 사이트로 구성됩니다.

추가 정보:

Magic number :
해당 없음

파일 확장자 :

.did

Macintosh 파일 유형 코드 :

TEXT

추가 정보를 위해 연락할 사람 및 이메일 주소 :

Ivan Herman 〈ivan@w3.org〉

용도 :

일반

사용 제한 :

없음

저자 :

Drummond Reed, Manu Sporny, Markus Sabadello, Dave Longley, Christopher Allen, Jonathan Holt

컨트롤러 변경 :

W3C

application / did + cbor와 함께 사용되는 프리그먼트 식별자는 DID Core v1.0, Fragment [DID-CORE]에 정의된 규칙에 따라 처리됩니다.

B.4 응용 프로그램 / did + dag + cbor

유형 이름 :
application

하위 유형 이름 :
did + dag + cbor

필수 매개 변수 :
없음

선택적 매개 변수 :
없음

인코딩 고려 사항 :
RFC 7049, 섹션 4.2를 참조하십시오.

보안 고려 사항 :
RFC 7049, 섹션 10 [RFC7049]를 참조하십시오.

상호 운용성 고려 사항 :
해당 없음

게시된 사양 :
http://www.w3.org/TR/did-core/

이 미디어 유형을 사용하는 응용 프로그램 :
분산되고 영구적이며 암호화로 검증 가능하고 확인 가능한 식별자가 필요한 모든 애플리케이션. 애플리케이션은 일반적으로 암호화 ID 시스템, 분산형 장치 네트워크, W3C 검증 가능한 자격 증명을 발급하거나 검증하는 웹 사이트로 구성됩니다.

추가 정보 :

Magic number :
해당 없음

파일 확장자 :

.did

Macintosh 파일 유형 코드 :

TEXT

추가 정보를 위해 연락할 사람 및 이메일 주소 :

Ivan Herman 〈ivan@w3.org〉

용도 :

일반

사용 제한 :

없음

저자 :

Drummond Reed, Manu Sporny, Markus Sabadello, Dave Longley, Christopher Allen, Jonathan Holt

컨트롤러 변경 :

W3C

application / did + cbor와 함께 사용되는 프리그먼트 식별자는 DID Core v1.0, Fragment [DID-CORE]에 정의된 규칙에 따라 처리됩니다.

C. 참고 문헌

C.1 Normative referances

[BASE58]

Base58 Encoding Scheme. Manu Sporny. IETF. 2019년 12월 인터넷 초안. URL : https://tools.ietf.org/html/draft-msporny-base58

[DID-CORE]

Decentralized Identifiers (DID) v1.0. Drummond Reed; Manu Sporny; Markus Sabadello; Dave Longley; Christopher Allen; Jonathan Holt. W3C. 2020년 7월 31일. W3C 작업 초안. URL : https://www.w3.org/TR/did-core/

[DID-SPEC-REGISTRIES]

DID Specification Registries. Orie Steele; Manu Sporny. Decentralized Identifier 워킹 그룹. W3C 편집자 초안. URL : https://w3c.github.io/did-spec-registries/

[INFRA]

Infra Standard. Anne van Kesteren; Domenic Denicola. WHATWG. Living Standard. URL : https://infra.spec.whatwg.org/

[JSON-LD]

JSON-LD 1.0. Manu Sporny; Gregg Kellogg; Markus Lanthaler. W3C. 2014년 1월 16일. W3C 권장 사항. URL : https://www.w3.org/TR/json-ld/

[JSON-LD11]

JSON-LD 1.1. Gregg Kellogg; Pierre-Antoine Champin; Dave Longley. W3C. 2020년 7월 16일. W3C 권장 사항. URL : https://www.w3.org/TR/json-ld11/

[RFC2119]

Key words for use in RFCs to Indicate Requirement Levels. S. Bradner. IETF. 1997년 3월. Best Current Practice. URL : https://tools.ietf.org/html/rfc2119

[RFC3552]

Guidelines for Writing RFC Text on Security Consideration. E. Rescorla; B. Korver.

IETF. 2003년 7월. Best Current Practice. URL : https://tools.ietf.org/html/rfc3552

[RFC3986]

URI (Uniform Resource Identifier) : Generic Syntax. T. Berners-Lee; R. Fielding; L. Masinter. IETF. 2005년 1월. 인터넷 표준. URL : https://tools.ietf.org/html/rfc3986

[RFC5234]

Augmented BNF for Syntax Specifications : ABNF. D. Crocker, Ed .; P. Overell. IETF. 2008년 1월. 인터넷 표준. URL : https://tools.ietf.org/html/rfc5234

[RFC6973]

Privacy Considerations for Internet Protocals. A. Cooper; H. Tschofenig; B. Aboba; J. Peterson; J. Morris; M. Hansen; R. Smith. IETF. 2013년 7월. 정보 제공. URL : https://tools.ietf.org/html/rfc6973

[RFC7049]

Concise Binary Object Representation (CBOR). C. Bormann; P. Hoffman. IETF. 2013년 10월. 제안된 표준. URL : https://tools.ietf.org/html/rfc7049

[RFC7517]

JSON Web Key (JWK). M. Jones. IETF. 2015년 5월. 제안된 표준.
URL : https://tools.ietf.org/html/rfc7517

[RFC7638]

JSON Web Key (JWK) Thumbprint. M. Jones; N. Sakimura. IETF. 2015년 9월. 제안된 표준. URL : https://tools.ietf.org/html/rfc7638

[RFC8141]

Uniform Resource Name (URN). P. Saint-Andre; J. Klensin. IETF. 2017년 4월. 제안된 표준. URL : https://tools.ietf.org/html/rfc8141

[RFC8152]

CBOR Object Signing and Encryption (COSE). J. Schaad. IETF. 2017년 7월. 제안된 표준. URL : https://tools.ietf.org/html/rfc8152

[RFC8174]
Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words. B. 레이 바. IETF. 2017년 5월. Best Current Practice. URL : https://tools.ietf.org/html/rfc8174

[RFC8259]
The JavaScript Object Notation (JSON) Data Interchange Format. T. Bray, Ed .. IETF. 2017년 12월 인터넷 표준. URL : https://tools.ietf.org/html/rfc8259

[RFC8610]
Concise Data Definition Language (CDDL) : A Notational Convention to Express Concise Binary Object Representation (CBOR) and JSON Data Structures. H. Birkholz; C. Vigano; C. Bormann. IETF. 2019년 6월. 제안된 표준. URL : https://tools.ietf.org/html/rfc8610

[XMLSCHEMA11-2]
W3C XML Schema Definition Language (XSD) 1.1 Part 2: Datatypes. David Peterson; Sandy Gao; Ashok Malhotra; Michael Sperberg-McQueen; Henry Thompson; Paul V. Biron et al. W3C. 2012년 4월 5일. W3C 권장 사항. URL : https://www.w3.org/TR/xmlschema11-2/

C.2 Informative references

[DID-RESOLUTION]
Decentralized Identifier Resolution. Markus Sabadello; Dmitri Zagidulin. Credentials Community Group. 커뮤니티 그룹 보고서 초안. URL : https://w3c-ccg. github.io/did-resolution/

[DID-RUBRIC]
Decentralized Characteristics Rubric v1.0. Joe Andrieu. Credentials Community Group 커뮤니티 그룹 보고서 초안. URL : https://w3c.github.io/did-rubric/

[DID-USE-CASES]
Decentralized Identifier Use Cases. Joe Andrieu; Kim Hamilton Duffy; Ryan Grant; Adrian Gropper. Decentralized Identifier Working Group. W3C 편집자 초안. URL :

https://w3c.github.io/did-use-cases/

[DNS-DID]
The Decentralized Identifier (DID) in the DNS. Alexander Mayrhofer; Dimitrij Klesev; Markus Sabadello. 2019년 2월. 인터넷 초안. URL : https://datatracker.ietf.org/doc/draft- mayrhofer-did-dns/

[HASHLINK]
Cryptographic Hyperlinks. Manu Sporny. IETF. 2018년 12월 인터넷 초안. URL : https://tools.ietf.org/html/draft-sporny-hashlink-02

[IANA-URI-SCHEMES]
Uniform Resource Identifier (URI) Schemes. IANA. URL : https://www.iana.org/assignments/uri-schemes/uri-schemes.xhtml

[MATRIX-URIS]
Matrix URIs-Ideas about Web Architecture. Tim Berners-Lee. 1996년 12월. Personal View. URL : https://www.w3.org/DesignIssues/MatrixURIs.html

[RFC4122]
A Universally Unique IDentifier (UUID) URN Namespace. P. Leach; M. Mealling; R. Salz. IETF. 2005년 7월. 제안된 표준. URL : https://tools.ietf.org/html/rfc4122

[RFC4648]
The Base16, Base32 및 Base64 Data Encodings. S. Josefsson. IETF. 2006년 10월. 제안된 표준. URL : https://tools.ietf.org/html/rfc4648

[RFC6901]
JavaScript Object Notation (JSON) Pointer. P. Bryan, Ed .; K. Zyp; M. Nottingham, Ed.. IETF. 2013년 4월. 제안된 표준. URL : https://tools.ietf.org/html/rfc6901

[RFC7230]
하이퍼 텍스트 전송 프로토콜 (HTTP / 1.1) : 메시지 구문 및 라우팅. R. Fielding, Ed .; J. Reschke, Ed.. IETF. 2014년 6월. 제안된 표준. URL : https://httpwg.org/specs/rfc7230.html

[RFC7231]

Hypertext Transfer Protocol (HTTP / 1.1) : Message Syntax and Routing. R. Fielding, Ed .; J. Reschke, Ed .. IETF. 2014년 6월. 제안된 표준. URL : https://httpwg.org/specs/rfc7231.html

[RFC7515]

JSON Web Signature(JWS). M. Jones; J. Bradley; N. Sakimura. IETF. 2015년 5월. 제안된 표준. URL : https://tools.ietf.org/html/rfc7515

[VC-DATA- MODEL]

Verifiable Credentials Data Model 1.0. Manu Sporny; Grant Noble; Dave Longley; Daniel Burnett; Brent Zundel. W3C. 2019년 11월 19일. W3C 권장 사항. URL : https://www.w3.org/TR/vc-data-model/

1. 본 「W3C 탈중앙ID 표준해설서」는 정부(과학기술정보통신부)의 재원으로, 정보통신기획평가원의 지원을 받은 과제(2017-0-00060, 사실표준화기구 전략대응 및 국제표준화전문가 활동강화)연구결과로 발간된 자료입니다.

◪ 총 괄 책 임 자 : 구경철(TTA 표준화본부장)
◪ 사 업 책 임 자 : 김대중(TTA 표준기획단장)
◪ 작성 및 검수자 : 정창진(플레이키키), 정상권((주)조이펀)
◪ 표 준 기 획 단 : 김영재, 김정헌, 전철기, 진수경, 전지윤, 임영선, 오지훈

블록체인 기반의 안전한 인증 및 데이터 주권을 위한

W3C 탈중앙ID 표준해설서 (Ver.2020)

2020년도 12월 인쇄
2020년도 12월 발행

발 행 소 : 한국정보통신기술협회
발 행 인 : 최 영 해
발간번호 : TTA-20114-SD
인 쇄 인 : (주)명진씨엔피 (02-2164-3000)

TTA 한국정보통신기술협회
Telecommunications Technology Association

463-824, 경기도 성남시 분당구 분당로 47
Tel : 031-780-9178, Fax : 031-724-0109
http://www.tta.or.kr