去中心化标识符 (DID) v1.1

核心架构、数据模型和表示法

W3C Editor's Draft 27 August 2024

▼ **More details about this document**

This version:
https://w3c.github.io/did-core/

Latest published version:
https://www.w3.org/TR/did-core-1.1/

Latest editor's draft:
https://w3c.github.io/did-core/

History:
Commit history

Editors:
Manu Sporny (Digital Bazaar) (v1.0, v1.1)
Dmitri Zagidulin (Invited Expert) (v1.0)
Steve McCown (v1.0)

Former editors:
Amy Guy (Digital Bazaar) (v1.0)
Markus Sabadello (Danube Tech) (v1.0)
Drummond Reed (Evernym/Avast) (v1.0)

Authors:
Manu Sporny (Digital Bazaar)
Dave Longley (Digital Bazaar)
Markus Sabadello (Danube Tech)
Drummond Reed (Evernym/Avast)
Orie Steele (Transmute)
Christopher Allen (Blockchain Commons)

Feedback:
GitHub w3c/did-core (pull requests, new issue, open issues)
public-did-wg@w3.org with subject line `[did-core-1.1] … message topic …` (archives)

Related Documents
DID Use Cases and Requirements
DID Specification Registries
DID Core Implementation Report

摘要

去中心化标识符（DID）是一种新型标识符，支持可验证的、去中心化的数字身份。一个 DID 可以标识由 DID 控制者决定的任何主题（例如个人、组织、事物、数据模型、抽象实体等）。与典型的联合标识符相比，DIDs的设计使其分离于集中式注册表、身份提供者以及证书颁发机。具体来说，虽然可借助第三方协助发现与DID相关的信息，但该设计允许DID控制者无需任何第三方授权的情况下即可证明对其的控制。DIDs 是将 DID 主题与DID 文档相关联的URIs，允许与该主题相关联的可信交互。

每个 DID 文档都可以表达加密材料、验证方法或服务，提供了一组让DID控制者能够证明对 DID 控制的机制。服务可实现与 DID 主题相关的可信交互。如果 DID 主题是数据模型等信息资源，则 DID 可提供返回 DID 主题本身的方法。

本文档规定了 DID 语法、通用数据模型、核心属性、序列化表示、DID 操作，并解释了将 DID 解析为其所代表的资源的过程。

本文现状

本节介绍本文档出版时的状态。当前 W3C 出版物和本技术报告最新修订版的列表可在 https://www.w3.org/TR/ 的 W3C 技术报告索引中找到。

此版本的 DID 核心标准（1.1 版）是试验性的。请勿实施。如果要实施 DID，请使用当前的 1.0 版标准： Decentralized Identifiers (DIDs) v1.0.

本文档由 Decentralized Identifier Working Group 作为编辑草案发布。

作为编辑草案发布并不意味着得到 W3C 及其成员的认可。

本文件为草案，可能随时被其他文件更新、取代或废止。除正在进行的工作外，不宜引用本文件。

本文档由一个根据 W3C 专利政策运作的小组编写。W3C 保留一份公开清单，记录与该小组交付成果相关的任何专利披露；该页面

还包括披露专利的说明。如果个人实际知晓一项专利，并认为该专利包含必要权利要求，则必须根据 W3C 专利政策第 6 节披露该信息。

本文件受 2023 年 11 月 3 日 W3C 流程文件管辖。

1. 导言

本节为非规范性内容。

作为个人和组织，我们中的许多人都在各种场合使用全球唯一标识符。它们可用作通信地址（电话号码、电子邮件地址、社交媒体上的用户名）、身份证号码（护照、驾照、纳税 ID、医疗保险）和产品标识符（序列号、条形码、RFIDs）。URIs（统一资源标识符）用于网络资源，您在浏览器中浏览的每个网页都有一个全球唯一的 URL（统一资源定位符 `Uniform Resource Locator`）。

这些全球唯一标识符绝大多数不受我们控制。它们是由外部机构发布的，这些机构决定它们所指的是谁或什么，以及何时可以撤销。它们只在某些情况下有用，只被某些机构认可，而不是由我们选择。它们可能会随着组织的倒闭而消失或失效。它们可能泄露个人信息。在许多情况下，它们可能会被恶意的第三方以欺诈方式复制和利用，这就是通常所说的“身份盗窃”。

本规范定义的去中心化标识符（DID）是一种新型的全球唯一标识符。其目的是让个人和组织能够使用他们信任的系统生成自己的标识符。这些新标识符使实体能够通过使用数字签名等加密证明进行验证，从而证明对标识符的控制权。

由于去中心化标识符的生成和声明是由实体控制的，每个实体都可以根据需要拥有尽可能多的去中心化标识符，以保持其所需的身份、角色和互动的分离。这些标识符的使用范围可以根据不同的情况进行适当调整。它们支持与其他人、机构或系统的交互，这些交互要求实体识别自己或自己控制的事物，同时提供对个人隐私数据披露程度的控制，而所有这些都不依赖于中央机构来保证标识符的持续存在。DID 用例文档 [DID-USE-CASES] 对这些想法进行了探讨。

本规范并不预设任何特定的技术或加密技术来支持 DID 的生成、持久性、解析或解释。例如，实施者可以根据在联合或中心化身份管理系统中注册的标识符创建去中心化标识符。事实上，几乎所有类型的标识符系统都可以增加对 DID 的支持。这就在中心化、联合式和去中心化标识符之间架起了一座互操作性桥梁。这也使实施者能够设计特定类型的 DID，以便与他们信任的计算基础设施（如分布式账本、去中心化文件系统、分布式数据库和点对点网络）协同工作。

本规范适用于：

- 希望了解作为去中心化标识符基础的核心架构原则的任何人；
- 希望生产和使用去中心化标识符及其相关数据格式的软件开发人员；
- 希望了解如何在其软件和硬件系统中使用去中心化标识符的系统集成商；
- 希望创建符合本文档所述生态系统的新 DID 基础设施（称为 DID 方法）的标准作者。

除本标准外，读者可能会发现去中心化标识符的用例和要求 [DID-USE-CASES] 文档也很有用。

1.1 一个简单的例子

本节为非规范性内容。

DID 是一个简单的文本字符串，由三部分组成：

1）DID URI 方案标识符；（`did URI scheme identifier`）

2）DID 方法标识符；（`DID method`）

3）DID 方法特定标识符。（`DID method-specific identifier`）

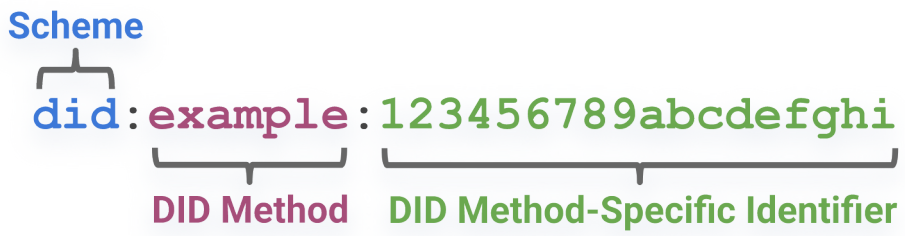

图 1 分散式标识符 (DID) 的简单示例

上面的 DID 示例解析为一个 DID 文档。DID 文档包含与 DID 相关的信息，如对 DID 控制器进行加密验证的方法。

> 示例 1: 一个简单的DID文档

```
{
  "@context": [
    "https://www.w3.org/ns/did/v1",
    "https://w3id.org/security/suites/ed25519-2020/v1"
  ],
  "id": "did:example:123456789abcdefghi",
  "authentication": [{
    // used to authenticate as did:...fghi
    "id": "did:example:123456789abcdefghi#keys-1",
    "type": "Ed25519VerificationKey2020",
    "controller": "did:example:123456789abcdefghi",
    "publicKeyMultibase": "z6MkmM42vxfqZQsv4ehtTjFFxQ4sQKS2w6WR7emozFAn5cxu"
  }]
}
```

1.2 设计目标

本节为非规范性内容。

去中心化标识符是更大系统的一个组成部分，如可验证凭证生态系统 [VC-DATA-MODEL]，这影响了本规范的设计目标。这里总结了去中心化标识符的设计目标。

目标	描述
去中心化	在标识符管理（包括全球唯一标识符、公共验证密钥、服务和其他信息的注册）中消除对中心化机构或单点故障的要求。
控制权	赋予实体（包括人类和非人类实体）直接控制其数字标识符的权力，而无需依赖外部机构。
隐私	使实体能够控制其信息的隐私，包括最小化、选择性和渐进式地披露属性或其他数据。
安全性	为请求方提供足够的安全性，使其能够依赖 DID 文档来获得所需的保证级别。
基于证明	让 DID 控制者在与其他实体交互时提供加密证明。
可发现性	使实体可以发现其他实体的 DID，以了解更多信息或与这些实体进行交互。
互操作性	使用互操作性标准，使 DID 基础设施可以利用为实现互操作性而设计的现有工具和软件库。
可移植性	与系统和网络无关，使实体能够在任何支持 DID 和 DID 方法的系统中使用其数字标识符。
简便性	简化功能，使技术更易于理解、实施和部署。
可扩展性	只要不严重妨碍互操作性、可移植性或简易性，应尽可能实现可扩展性。

1.3 架构概述

本节为非规范性内容。

本节提供去中心化标识符架构主要组件的基本概述。

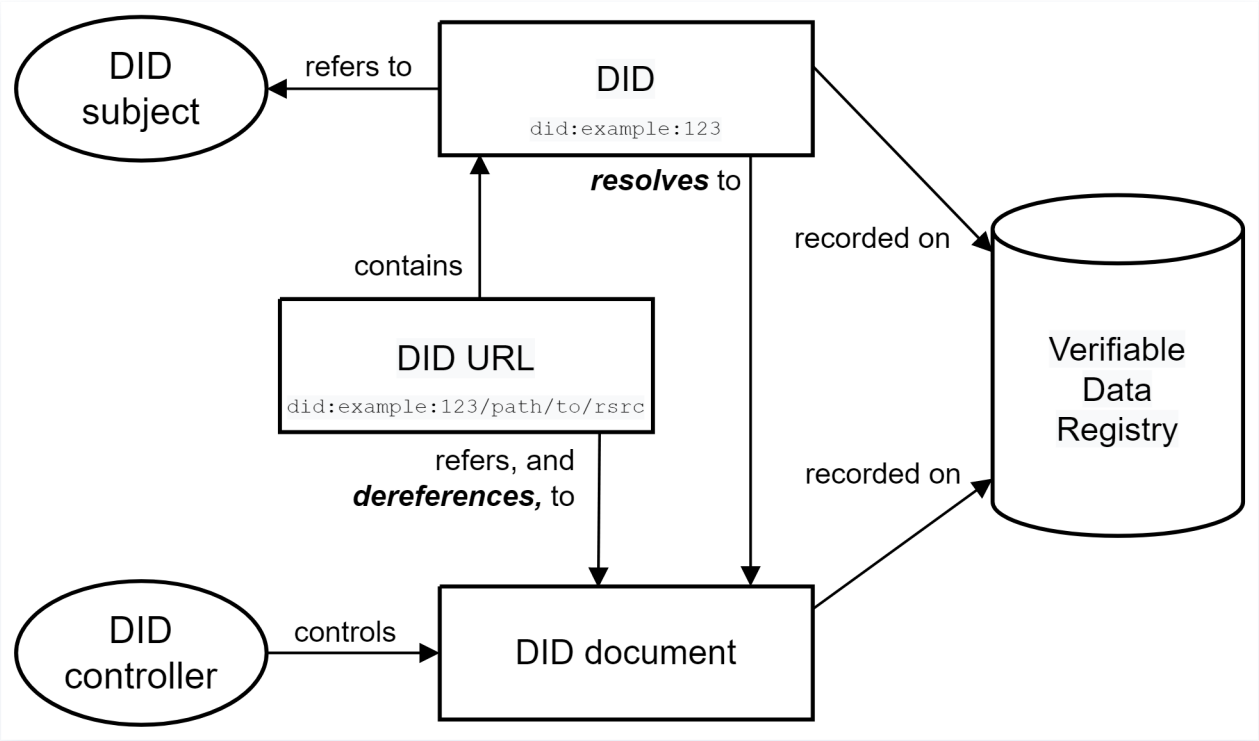

图 2 DID 架构概览和基本组件之间的关系。另见：叙述性说明。

DIDs 和 DID URLs

去中心化标识符或 DID 是由三部分组成的 URI：方案 did:(`scheme did:`)、方法标识符(`method identifier`)和 DID 方法(`DID method`)指定的唯一特定方法标识符。DID 可解析为 DID 文档(`DID documents`)。DID URL 扩展了基本 DID 的语法，纳入了路径、查询和片段等其他标准 URI 组件，用于定位特定资源--例如，DID 文档中的加密公钥或 DID 文档外部的资源。这些概念将在3.1 DID 语法和3.2 DID URL 语法中详细阐述。

DID 主题

DID 主题顾名思义就是 DID 所标识的实体。DID 主题也可以是 DID 控制器。任何东西都可以是 DID 的主题：人、团体、组织、事物或概念。5.1.1 DID 主题对此有进一步定义。

DID 控制者

DID 控制者是指有能力（如 DID 方法所定义）对 DID 文档进行更改的实体（个人、组织或自主软件）。这种能力通常由代表控制者的软件对一组加密密钥的控制来实现，但也可能通过其他机制来实现。请注意，一个 DID 可能有不止一个控制器，而 DID 主题可以是 DID 控制器，也可以是其中之一。这一概念在 5.1.2 DID 控制器中有所阐述。

可验证的数据注册

为了能解析 DID 文档，DID 通常会记录在某种底层系统或网络中。无论使用何种特定技术，任何支持记录 DID 并返回生成 DID 文档所需数据的系统都称为可验证数据注册中心。这方面的例子包括分布式分类账、去中心化文件系统、各种数据库、点对点网络以及其他形式的可信数据存储。这一概念将在8. 方法中进一步阐述。

DID 文档

DID 文档包含与 DID 相关的信息。它们通常表达与 DID 主题交互相关的验证方法（如加密公钥）和服务。DID 文档支持的通用属性在5.核心属性中。DID 文档可序列化为字节流（见 6.表示法）。DID 文档中的属性可根据 8.方法。

DID 方法

DID 方法是创建、解析、更新和停用特定类型 DID 及其关联 DID 文档的机制。DID 方法使用单独的 DID 方法规范来定义，如8. 方法中的定义。

DID 解析器和 DID 解析

DID 解析器是一个系统组件，它将 DID 作为输入，并将符合要求的 DID 文档作为输出。这一过程称为 DID 解析。解析特定类型 DID 的步骤由相关 DID 方法规范定义。DID 解析过程详见 7. 解析。

DID URL 解除引用器和 DID URL 解除引用

DID URL 解除引用器是一个将 DID URL 作为输入并将资源作为输出的系统组件。这一过程称为 DID URL 解除引用。7.2 DID URL 解除引用中详细介绍了 DID URL 解除引用的过程。

1.4 一致性

除标注为非规范性的部分外，本规范中的所有编写指南、图表、示例和注释都是非规范性的。本规范中的其他内容均为规范性内容。

本文档中的关键词 MAY、MUST、MUST NOT、OPTIONAL、RECOMMENDED、REQUIRED、SHOULD 和 SHOULD NOT，当且仅当它们以大写字母出现时，应按照 BCP 14 [RFC2119] [RFC8174] 中的描述进行解释，如此处所示。

本文档包含包含 JSON 和 JSON-LD 内容的示例。其中一些示例包含无效字符，例如内联注释 (//) 和使用省略号（...）来表示对示例没有什么价值的信息。如果实施者希望将这些信息用作有效的 JSON 或 JSON-LD，请务必删除这些内容。

某些示例包含本规范未定义的术语（包括属性名称和属性值）。这些术语用注释（//外部（属性名|值））表示。这些术语在 DID 文档中使用时，应在 DID 规范注册表 [DID-SPEC-REGISTRIES] 中注册，并链接到正式定义和 JSON-LD 上下文。

测试DID 和 DID 文档实现的互操作性是通过评估创建和解析符合本规范的 DID 和 DID 文档的能力。DID 和 DID 文档生产者和解析器的互操作性是通过确保 DID 和 DID 文档符合规范来实现的。DID 方法规范的互操作性参考每个 DID 方法规范中的具体细节。例如，像网络浏览器不需要实现所有已知的 URI 方案一样，与 DID 兼容的软件也不需要实现所有已知的 DID 方法。但是，特定 DID 方法的所有实现都应具有互操作性。

符合要求的 DID 是对 3.标识符 中规定的规则在符合该节中的相关规范声明下的具体实现。

符合要求的 DID 文档是本规范中描述的数据模型在符合 4. 数据模型 和 5. 核心属性 下的实现。符合要求的文档的序列化格式是确定的、双向的和无损的，如 6. 表述 。

符合规范的生产者是以软件和/或硬件形式实现的任何算法，用于在符合 6.表述 中的相关规范声明下生成符合规范的 DID 或符合标准的 DID 文档。

符合标准的解析器是指作为软件和/或硬件实现的任何算法，它解析符合标准的 DID 或符合标准的 DID 文档，并遵守 6.表述 中的相关规范声明。

符合标准的 DID 解析器是指在符合 7.1 DID 解析中的相关规范声明下任何以软件和/或硬件形式实现的算法。

符合要求的 DID URL 解除器是指以软件和/或硬件形式实现的、符合 7.2 DID URL 解除中相关规范声明的任何算法。

符合标准的 DID 方法是指符合 8.方法中相关规范声明的任何规范。

2. 术语

本节为非规范性内容。

本节定义了本标准和整个去中心化标识符基础中使用的术语。只要这些术语在本标准中出现，就会提供相关链接。

放大攻击（`amplification attack`）
一类攻击，攻击者通过向系统提供少量有效输入，试图耗尽目标系统的 CPU、存储、网络或其他资源，从而造成破坏性影响，其处理成本可能以指数形式超过输入本身。

验证（`authenticate`）
身份验证是一个实体使用一种或多种验证方法证明其拥有特定属性或控制特定秘密的过程。就 DID 而言，一个常见的例子是证明与 DID 文档中公布的公钥相关的加密私钥的控制权。

加密套件（`cryptographic suite`）
为实现特定安全目标而定义特定加密原语用法的标准。这些文件通常用于指定验证方法、数字签名类型、其标识符和其他相关属性。

去中心化识符（`decentralized identifier (DID)`）
全球唯一永久的标识符，不需要中央注册机构，通常以加密方式生成和/或注册。3.1 DID 语法中定义了 DID 的通用格式。具体的 DID 方案在 DID 方法标准中定义。许多——但并非所有——DID 方法都使用分布式账本技术（DLT）或其他形式的去中心化网络。

去中心化身份管理（`decentralized identity management`）
基于使用去中心化标识符为的身份管理。将生成、注册和分配标识符的权力扩展到传统的信任根基之外，如 X.500 目录服务、域名系统和大多数国家的身份证系统。

DID 控制者（`DID controller`）
具有能力更改 DID 文档的实体。一个 DID 可能有不止一个 DID 控制器。DID 控制器可通过 DID 文档顶层的可选控制器属性表示。请注意，DID 控制器可能就是 DID 主题。

DID 委托人（`DID delegate`）

DID 控制员通过 DID 文档授权使用与 DID 相关的验证方法的实体。例如，控制孩子 DID 文档的家长可能允许孩子使用个人设备进行身份验证。在这种情况下，孩子就是 DID 委托人。孩子的个人设备包含私人加密材料，使孩子能够使用 DID 进行身份验证。不过，未经父母允许，孩子可能不得添加其他个人设备。

DID 文档（`DID document`）

描述 DID 主题的一组数据。其包括 DID 主题或 DID 委托人可用于验证自身身份并证明与 DID 关联的机制（如加密公钥）。一个 DID 文档可能有一个或多个不同的表示法，如 6.表示法或 W3C 文件中的定义。表示法或 W3C DID 标准注册表 [DID-SPEC-REGISTRIES] 中定义的一种或多种不同表示法。

DID 片段（`DID fragment`）

DID URL 中第一个散列符号字符 (#) 之后的部分。DID 片段语法与 URI 片段语法相同。

DID 方法（`DID method`）

特定 DID 方法方案实现方式的定义。DID 方法由 DID 方法标准规定，该标准规定了创建、解析、更新和停用 DID 和 DID 文档的详细操作。请参阅 8.方法 。

DID 路径（`DID path`）

DID URL 中以第一个正斜线 (/) 字符开头并包含该字符的部分，以问号 (?)、片段哈希符号 (#) 字符或 DID URL 结尾结束。DID 路径语法与 URI 路径语法相同。请参阅路径。

DID 查询（`DID query`）

DID URL 中第一个问号字符 (?) 之后的部分。DID 查询语法与 URI 查询语法相同。请参阅查询。

DID 解析（`DID resolution`）

将 DID 和一组解析选项作为输入，并以符合要求的表示形式返回 DID 文档和附加元数据的过程。该流程依赖于适用 DID 方法的 “读取 ”操作。该流程的输入和输出在 7.1 DID 解析中定义。

DID 解析器（`DID resolver`）

DID 解析器是执行 DID 解析功能的软件和/或硬件组件，它将 DID 作为输入，并将符合要求的 DID 文档作为输出。

DID 方案（`DID scheme`）

去中心化标识符的正式语法。通用 DID 方案以 3.1 DID 语法中定义的前缀 did: 开始。每个 DID 方法标准都定义了与特定 DID 方法配合使用的特定 DID 方法方案。在特定 DID 方法方案中，DID 方法名称以第一个冒号开头，以第二个冒号结尾，如 did:example:

DID 主题（`DID subject`）

由 DID 认证并由 DID 文档描述的实体。任何东西都可以成为 DID 主题：人、群体、组织、客观事物、数字事物、主观事物。

DID URL

DID 加上符合 3.2 DID URL 语法定义的任何附加语法组件。这包括可选的 DID 路径（带前导 / 字符）、可选的 DID 查询（带前导 ? 字符）和可选的 DID 片段（带前导 # 字符）。

DID URL 解引用（`DID URL dereferencing`）
将一个 DID URL 和一组输入元数据作为输入，并返回一个资源的过程。该资源可能是 DID 文档加上附加元数据，也可能是 DID 文档中包含的二级资源，还可能是与 DID 文档无关的资源。该流程使用 DID 解析来获取 DID URL 中包含的 DID 所指示的 DID 文档。解引用流程可对 DID 文档处理，以返回 DID URL 所指的解引用资源。该流程的输入和输出在 7.2 DID URL 解引用中定义。

DID URL 解引用器（`DID URL dereferencer`）
对给定 DID URL 或 DID 文档执行 DID URL 解引用功能的软件和/或硬件系统。

分布式账本（`distributed ledger (DLT)`）
用于记录事件的非集中式系统。这些系统为参与者建立了足够的信任，使其能够依赖他人记录的数据做出操作决策。它们通常使用分布式数据库，不同节点使用共识协议确认加密签名交易的排序。经过数字签名的交易随着时间的推移被连接起来，这通常会使账本的历史不可更改。

公钥描述（`public key description`）
包含在 DID 文档中的数据对象，其中包含使用公钥或验证密钥所需的所有元数据。

资源（`resource`）
由 [RFC3986] 定义："……术语'资源'用于一般意义上的任何可由 URI 标识的内容"。同样，任何资源都可以作为由 DID 标识的 DID 主题。

表示法（`representation`）
正如 [RFC7231] 为 HTTP 所定义："旨在反映指定资源的过去、当前或期望状态的信息，其格式可通过协议轻松通信，由一组表示元数据和可能无限制的表示数据流组成"。DID 文档是描述 DID 主题的信息表征。见 6. 表示。

特殊表征条目（`representation-specific entries`）
DID 文档中的条目，其含义是特定表示法所特有的。在 4. 数据模型》和《6. 表示法中定义。例如，JSON-LD 表示法中的 @context 就是特定于表示法的条目。

服务（`services`）
通过一个或多个服务端点与 DID 主题或相关实体进行通信或交互的方式。例如，发现服务、代理服务、社交网络服务、文件存储服务和可验证凭证存储库服务。

服务端点（`service endpoint`）
一个服务代表 DID 主题运行的网络地址，如 HTTP URL。

统一资源标识符（`Uniform Resource Identifier (URI)`）
由 [RFC3986] 定义的万维网上所有资源的标准标识符格式。DID 是 URI 方案的一种。

可验证凭证（`verifiable credential`）
W3C 可验证凭证规范 [VC-DATA-MODEL] 所定义的加密可验证数字凭证的标准数据模型和表示格式。

可验证数据注册中心（`verifiable data registry`）
一种便于创建、验证、更新和/或停用去中心化标识符和 DID 文档的系统。可验证数据注册中心也可用于其他加密可验证数据结构，如可验证凭证。更多信息，请参阅 W3C 可验证凭证规范 [VC-DATA-MODEL]。

可验证时间戳（`verifiable timestamp`）
可验证时间戳可让第三方验证数据对象在某一特定时刻是否存，以及自该时刻起是否被修改或损坏。如果数据完整性从该时刻起已被修改或损坏，则时间戳无法验证。

验证方法（`verification method`）
一组参数，可与流程一起用于独立验证证明。例如，加密公共密钥可用作数字签名的验证方法；在这种用法中，它可以验证签名者是否拥有相关的加密密钥。

本定义中的 "验证 "和 "证明 "意在广泛适用。例如，在 Diffie-Hellman 密钥交换过程中，可以使用加密公开密钥来协商加密共享对称密钥。这保证了密钥协议过程的完整性。因此，这也是另一种验证方法，尽管对这一过程的描述可能不会使用 "验证 "或 "证明 "等词。

验证关系（`verification relationship`）
DID 主题与验证方法之间关系的表述。5.3.1 验证就是验证关系的一个例子。

全球唯一标识符（`Universally Unique Identifier (UUID)`）
由 [RFC4122] 定义的一种全球唯一标识符。UUID 与 DID 类似，都不需要集中注册机构。UUID 与 DID 的不同之处在于它们不可解析或不可加密验证。
除上述术语外，本标准还使用 [INFRA] 规范中的术语来正式定义数据模型。当使用 [INFRA] 术语（如字符串、集合和映射）时，会直接链接到该标准。

3. 标识符

本节介绍 DID 和 DID URL 的语法。术语 "通用 "是为了将此处定义的语法与特定 DID 方法在各自规范中定义的语法区分开来。DID 和 DID URL 的创建过程及其时间在 8.2 方法操作和 B.2 创建 DID 中描述。

3.1 DID 语法

通用 DID 方案是一种符合 [RFC3986] 的 URI 方案。ABNF 定义如下，它使用 [RFC5234] 中的语法以及 ALPHA 和 DIGIT 的相应定义。ABNF 中未定义的所有其他规则名称均在 [RFC3986] 中定义。所有 DID 必须符合 DID 语法 ABNF 规则。

DID 语法 ABNF 规则

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

有关 DID 方法与 DID 语法相关的要求，请参见第 8.1 节 方法语法。

3.2 DID URL 语法

DID URL 是特定资源的网络位置标识符。它可用于检索 DID 主题的表示、验证方法、服务、DID 文档的特定部分或其他资源。

以下是使用 [RFC5234] 中语法的 ABNF 定义。它基于以 3.1 DID 语法中定义的 did 方案。路径空、查询和片段组件在 [RFC3986] 中定义。所有 DID URL 必须符合 DID URL 语法 ABNF 规则。如 8.1 方法语法所述，DID 方法可以进一步限制这些规则。

DID URL 语法 ABNF 规则

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

> 注意：分号字符保留供将来使用
>
> 虽然分号 (;) 字符可根据 DID URL 语法标准使用，但本规范的未来版本可能会将其用作参数的子分隔符，如 [MATRIX-URIS] 所述。为避免将来发生冲突，开发人员应避免使用它。

路径

DID 路径与通用 URI 路径相同，并符合 RFC 3986 第 3.3 节中的路径空 ABNF 规则。与 URI 一样，路径语义可由 DID 方法指定，这反过来又可使 DID 控制器能够进一步细化这些语义。

> 示例 2
>
> `did:example:123456/path`

查询

DID 查询与一般 URI 查询相同，符合 RFC 3986 第 3.4 节中的查询 ABNF 规则。3.2.1 DID 参数将详细介绍这一语法特征。

> 示例 3
>
> `did:example:123456?versionId=1`

片段

DID 片段的语法和语义与通用 URI 片段相同，并符合 RFC 3986 第 3.5 节中的片段 ABNF 规则。

DID 片段可用作 DID 文档或外部资源的独立于方法的引用。下面是一些 DID 片段标识符的示例。

> 例 4：DID 文档中的唯一验证方法
>
> `did:example:123#public-key-0`

> 例 5：DID文档中的唯一服务端点
>
> `did:example:123#agent`

> 例 6：DID 文档中的外部资源
>
> `did:example:123?service=agent&relativeRef=/credentials#degree`

> 注：跨表述的片段语义
>
> 为了最大限度地提高互操作性，我们敦促实现者确保 DID 片段在不同表示法中的解释方式相同（请参阅 6.表示法）。例如，虽然可以在 DID 片段中使用 JSON 指针 [RFC6901]，但它在非 JSON 表示法中的解释方式并不相同。

E.2 application/did+ld+json 中针对 JSON-LD 表示法描述了片段标识符的其他语义，这些语义与本节中的语义兼容，并对其进行了分层。有关如何解引用 DID 片段的信息，请参阅 7.2 DID URL 解引用。

3.2.1 DID 参数

DID URL 语法支持基于 "查询 "中描述的查询组件的简单参数格式。在 DID URL 中添加 DID 参数意味着该参数成为资源标识符的一部分。

> 例 7：带有 "versionTime "DID 参数的 DID URL
> `did:example:123?versionTime=2021-05-10T17:00:00Z`

> 例 8：带有 "service "和 "relativeRef "DID 参数的 DID URL
> `did:example:123?service=files&relativeRef=/resume.pdf`

有些 DID 参数完全独立于任何特定的 DID 方法，对所有 DID 起相同的作用。其他 DID 参数并非所有 DID 方法都支持。在支持可选参数的情况下，这些参数在支持它们的DID方法中具有统一的操作方式。下表提供了所有 DID 方法中功能相同的常用 DID 参数。对所有 DID 参数的支持都是可选的。

> 注意
> 一般来说，DID URL 解引用器的实现应参考 [DID-RESOLUTION] 以了解更多实现细节。本标准的范围仅定义了最常见的查询参数的规定。

参数名称	描述
service	通过服务 ID 从 DID 文档中标识一项服务。如果存在，相关值必须是 ASCII 字符串。
relativeRef	根据 RFC3986 第 4.2 节规定的相对 URI 引用，用于标识服务端点的资源，该资源是通过使用服务参数从 DID 文档中选择的。如果存在，相关值必须是 ASCII 字符串，并且必须对某些字符使用 RFC3986 第 2.1 节规定的百分数编码。
versionId	标识要解析的 DID 文档的特定版本（版本 ID 可以是顺序、UUID 或特定方法）。如果存在，相关值必须是 ASCII 字符串。
versionTime	标识要解析的 DID 文档的特定版本时间戳。也就是说，该 DID 文档在特定时间对 DID 有效。如果存在，相关值必须是一个 ASCII 字符串，它是一个有效的 XML 日期值，如 W3C XML 模式定义语言（XSD）1.1 第 2 部分：数据类型 [XMLSCHEMA11-2] 第 3.3.7 节所定义。该日期时间值必须归一化为 UTC 00:00:00，并且不含小数点后两位。例如：2020-12-20T19:17:47Z。
hl	DID 文档的资源哈希值，用于添加完整性保护，如 [HASHLINK] 所规定。此参数不具规范性。如果存在，相关值必须是 ASCII 字符串。

实施者和 DID 方法规范作者可能会使用此处未列出的其他 DID 参数。为实现最大程度的互操作性，建议 DID 参数使用 DID 标准注册机制 [DID-SPEC-REGISTRIES]，以避免与具有不同语义的相同 DID 参数的其他用途发生冲突。

如果有明确的使用案例（需要将 DID 参数作为 URL 的一部分）使用DID参数是一种比单独使用DID更精确的引用方式。如果可以通过向 DID 解析器传递输入元数据来实现相同的功能，则可以不使用 DID 参数。有关处理这些参数的其他注意事项，请参阅 [DID-RESOLUTION]。

> 注意：DID 参数和 DID 解析
>
> 将不属于 DID URL 的输入元数据传递给 DID 解析器会影响 DID 解析和 DID URL 解引用功能（请参阅 7.1.1 DID 解析选项）。这与 HTTP 类似，某些参数可以包含在 HTTP URL 中，也可以在解引用过程中作为 HTTP 标头传递。重要的区别在于，作为 DID URL 一部分的 DID 参数应用于指定正在标识的资源，而非作为DID URL一部分的输入元数据应用于控制该资源的解析或解引用方式。

3.2.2 相对 DID URL

相对 DID URL 是 DID 文档中任何不以 `did:<method-name>:<method-specific-id>` 开头的 URL 值。更具体地说，它是指任何不以 3.1 DID 语法中定义的 ABNF 开头的 URL 值。该URL 应引用同一 DID 文档中的资源。相对 DID URL 可以包含相对路径组件、查询参数和片段标识符。

解析相对 DID URL 引用时，必须使用 RFC3986 第 5 节：引用解析中指定的算法。**基础 URI** 值是与 DID 主题相关联的 DID，请参阅 5.1.1 DID 主题。**方案**为 did。**权限**是 `<method-name>:<method-specific-id>` 的组合，**路径**、**查询**和**片段**值分别是路径、查询和片段中定义的值。

相对 DID URL 通常用于引用 DID 文档中的验证方法和服务，而无需使用绝对 URL。考虑到存储空间大小，可使用相对 URL 来减少存储 DID 文档的大小。

> 例 9：相对 DID URL 示例

```
{
  "@context": [
    "https://www.w3.org/ns/did/v1",
    "https://w3id.org/security/suites/ed25519-2020/v1"
  ]
  "id": "did:example:123456789abcdefghi",
  "verificationMethod": [{
    "id": "did:example:123456789abcdefghi#key-1",
```

```
      "type": "Ed25519VerificationKey2020", // external(property value)
      "controller": "did:example:123456789abcdefghi",
      "publicKeyMultibase": "z6MkmM42vxfqZQsv4ehtTjFFxQ4sQKS2w6WR7emozFAn5cxu"
    }, ...],
    "authentication": [
      // a relative DID URL used to reference a verification method above
      "#key-1"
    ]
  }
```

在上例中，相对 DID URL 值将转换为 `did:example:123456789abcdefghi#key-1` 的绝对 DID URL 值。

4 数据模型

本标准定义了一个数据模型，用于表达 DID 文档和 DID 文档数据结构，些结构可进一步序列化为多种具体表示形式。本节提供了数据模型的高级描述、数据模型中不同类型属性的表达方式描述以及扩展数据模型的说明。

DID 文档由条目映射组成，每个条目由键/值对组成。DID 文档数据模型至少包含两类不同的条目。第一类条目称为属性，在第 5 节核心属性中定义。第二类为表示特定条目，在第6节表示中定义。

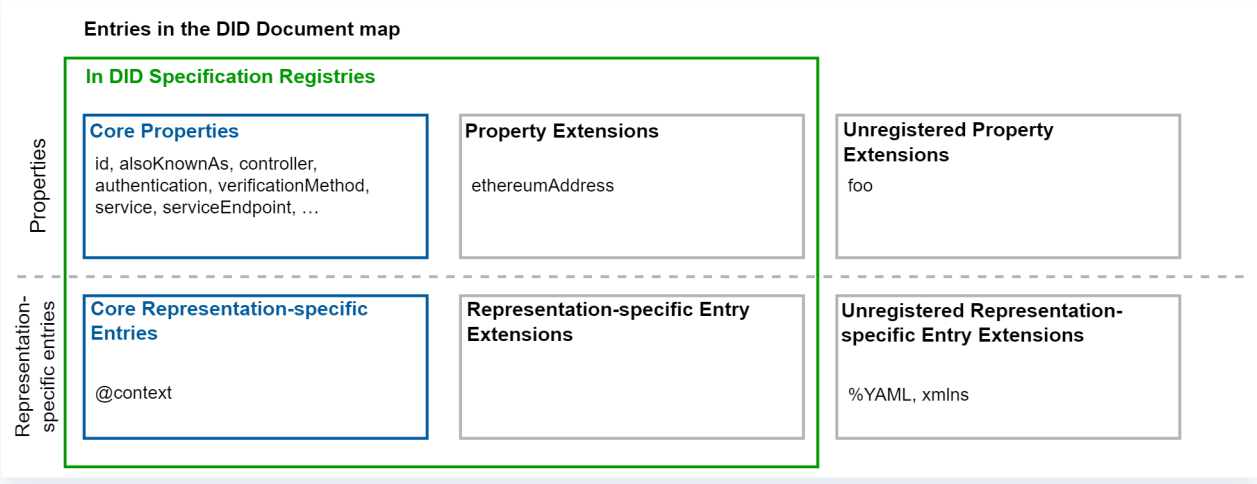

图 3 DID 文档中的条目。另请参阅：叙述性说明。

DID 文档数据模型中的所有条目键都是字符串。所有条目值都使用下表中的一种抽象数据类型表示，每种表示法都指定了每种数据类型的具体序列化格式。

数据类型注意事项

数据类型注意事项	
map	映射中规定的键/值对的有限有序序列，且键不会出现两次，如[INFRA]中所述。在 [INFRA] 中，映射有时被称为有序映射。
list	[INFRA]中规定的有限有序序列。
set	[INFRA]中规定的不重复包含同一项目的有限有序项目序列。在 [INFRA] 中，集合有时被称为有序集合。
datetime	日期和时间值，能够无损地表达 [XMLSCHEMA11-2] 中规定的 dateTime 可以表达的所有值。
string	字符串 [INFRA] 中规定的、通常用于表示人类可读语言的代码单元序列。
integer	整数 [XMLSCHEMA11-2] 中规定的不含分数成分的实数。为最大限度地提高互操作性，建议实施者注意 RFC8259 第 6 节：数字中有关整数的建议。
double	double 通常用于近似 [XMLSCHEMA11-2] 中规定的任意实数的值。为最大限度地提高互操作性，建议实施者注意 RFC8259 第 6 节：数字中有关 double 的建议。
boolean	按[INFRA] 中定义，该值要么为真，要么为假
null	按 [INFRA] 的定义，用于表示缺少值的值。

> 由于数据模型是使用 [INFRA] 中的术语定义的，因此可以包含多个项目的属性值（如列表、映射和集合）都是明确有序的。[INFRA]中所有类似列表的值结构都是有序的，无论这种有序是否重要。就本标准而言，除非另有说明，否则映射和集合的排序并不重要，也不希望实现产生或消耗确定排序的值。

4.1 可扩展性

数据模型支持两种类型的可扩展性。

1. 为获得为最大限度的互操作性，建议扩展使用 W3C DID 规范注册机制 [DID-SPEC-REGISTRIES]。对新属性或其他扩展使用该机制是确保两种不同表示法能够协同工作的唯一指定机制。
2. 表示可以定义其它可扩展性机制，包括不需要使用 DID 规范注册表的机制。此类扩展机制应支持无损转换成任何其他符合要求的表示法。表示法的扩展机制应定义所有属性和表示语法到数据模型及其类型系统的映射。

> 注意：未注册的扩展可靠性较低
> 两个特定实现之间可以始终达成共识，通过私下协商使用一种相互理解的扩展或表示形式，该形式未记录在 DID 规范注册中心 [DID-SPEC-REGISTRIES] 中；这种实现与更大生态系统的互操作性将降低可靠性。

5. 核心属性

DID 与 DID 文档关联。DID 文档使用数据模型表达，并可序列化为表示形式。以下章节定义 DID 文档中的属性，包括其是否为必选属性。这些属性描述了 DID 主题与属性值之间的关系。

下表列出了本标准定义的核心属性的参考信息，包括预期值和是否为必选。表中的属性名称链接到规范定义和每个属性的详细描述。

> 注意：不同类型映射中的属性名称
> 属性名称 id、type 和 controller 可能出现在不同类型的映射中，且约束条件可能有所差异。

DID 文件属性

属性	是否必选?	限制值
id	是	符合 3.1 DID 句法规则的字符串。
alsoKnownAs	否	符合[RFC3986]URI 规则的一组字符串。
controller	否	符合 3.1 DID 句法规则的一个字符串或一组字符串。
verificationMethod	否	符合验证方法属性中规则的一组验证方法映射。
authentication	否	
assertionMethod	否	一组符合验证方法属性规则的验证方法映射或符合 3.2 DID URL 语法的字符串集合。
keyAgreement	否	
capabilityInvocation	否	
capabilityDelegation	否	
service	否	符合服务属性中规则的一组服务端点映射。

验证方法属性

属性	是否必选?	限制值
id	是	符合 3.2 DID URL 语法规则的字符串。
controller	是	符合 3.1 DID 语法规则的字符串。
type	是	字符串
公钥JWK `publicKeyJwk`	否	表示符合 [RFC7517] 的 JSON 网络密钥的映射。有关其他限制，请参阅 publicKeyJwk 的定义。
公钥Multibase `publicKeyMultibase`	否	符合 [MULTIBASE] 编码公钥的字符串。

服务属性

属性	是否必选?	限制值
id	是	符合 [RFC3986] URI 规则的字符串。
type	是	字符串或一组字符串。 符合 [RFC3986] 中 URIs、映射或集合规则的字符串
serviceEndpoint	是	由一个或多个符合 [RFC3986] URIs 和/或映射规则的字符串组成。

5.1 标识符

本节描述 DID 文档如何包含 DID 主题和 DID 控制器的标识符。

5.1.1 DID 主题

特定 DID 主题的 DID 通过 DID 文档中的 id 属性表示。

- **id**

  id的值必须是符合 3.1 DID 语法规则的字符串，并且必须存在于 DID 文档的数据模型根映射中。

```
例10
{
  "id": "did:example:123456789abcdefghijk"
}
```

id 属性仅在 DID 文档的顶层映射中存在时才表示 DID 主题的 DID。

> **注意：中间表示**
>
> DID 方法标准可以创建不包含 id 属性的 DID 文档中间表示，例如在 DID 解析器执行 DID 解析时。然而，完全解析的 DID 文档始终包含有效的 id 属性。

5.1.2 DID 控制器

DID 控制器是授权更改 DID 文档的实体。授权 DID 控制器的过程由 DID 方法定义。

- **controller**

  controller 属性不是必选的，如果选择，其值必须是一个或多个符合 3.1 DID 语法规则的字符串。相应的 DID 文档应该包含明确允许使用特定验证方法进行特定目的的验证关系。

当 DID 文档中存在 controller 属性时，其值表示一个或多个 DID。这些 DID 的 DID 文档中包含的任何验证方法都应被视为已授权，因此满足这些验证方法的证明应被视为等同于 DID 主题提供的证明，并代表授权更新 DID 文档的 DID 控制器。

例11：带有控制器属性的 DID 文档

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

> **注意：授权与认证**
>
> 请注意，controller 值提供的授权与 5.3.1 认证中描述的认证是分开的。这对于密钥恢复尤其重要，例如在密钥丢失的情况下，DID 主题不再能访问其密钥，或者在密钥泄露的情况下，DID 控制器的可信第三方需要覆盖攻击者的恶意活动。有关威胁模型和攻击向量的详细信息，请参阅 9. 安全考虑。

5.1.3 别名

一个 DID 主题可以出于不同目的或在不同时间拥有多个标识符。可以使用 alsoKnownAs 属性来声明两个或多个 DID（或其他类型的 URI）引用同一个 DID 主题。

- **alsoKnownAs**

  alsoKnownAs 属性不是必选的，如果选择，其值必须是一个集合，其中每个项目都是符合 [RFC3986] 的 URI。

这种关系表示该标识符的主题也可以由一个或多个其他标识符标识。

> **注意：等价性和别名**
>
> 应用程序可以选择在 alsoKnownAs 关系在相反方向上被互换的情况下，将由 alsoKnownAs 关联的两个标识符视为等价。如果不具有这种逆向关系，最好不要将它们视为等价。换句话说，alsoKnownAs 断言的存在并不能证明该断言是真实的。因此，强烈建议请求方获得 alsoKnownAs 断言的独立验证。
>
> 鉴于DID主体可能出于不同目的使用不同的标识符，因此对这两种标识符之间的强等价性或合并两个相应DID文档的信息的期望，并不一定合适，即便存在相互关系。

5.2 验证方法

DID 文档可以表达验证方法，例如加密公钥，用于认证或授权与 DID 主题或相关方的交互。例如，加密公钥可以用作数字签名的验证方法；在这种用法中，它验证签名者可以使用关联的加密私钥。验证方法可能包含多个参数。例如，一组五个加密密钥中，任何三个都需要参与加密阈值签名。

- **verificationMethod**

  verificationMethod 属性不是必选的，如果选择，其值必须是一组验证方法，其中每个验证方法使用映射表示。 验证方法映射必须包含 id、type、controller 和由 type 值确定并在 5.2.1 验证材料中定义的特定验证材料属性。验证方法可以包含其他属性。验证方法应该在 DID 规范注册中心 [DID-SPEC-REGISTRIES] 中注册。

- **id**

  验证方法的 id 属性值必须是符合第 3.2 节 DID URL 语法规则的字符串。该值表示取消引用到已识别验证方法的 DID URL。

  - **type**

属性的值必须是引用一个验证方法类型的字符串。为了最大限度地实现全球互操作性，验证方法类型应该在 DID 规范注册中心 [DID-SPEC-REGISTRIES] 中注册。该值表示所使用的验证方法套件的类型。

- **controller**

  controller 属性的值必须是符合 3.1 DID 语法规则的字符串。该值表示拥有用于认证验证方法的秘密加密材料的 DID 控制器或 DID 代理。

例12：验证方法的结构

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

> **注意：验证方法控制器与 DID 控制器**
>
> 当关系主题是 DID 文档时，controller 属性的语义与关系主题是验证方法（例如加密公钥）时相同。由于密钥无法自我控制，并且密钥控制器无法从 DID 文档推断，因此需要明确表达密钥控制者的身份。不同之处在于，验证方法的 controller 值不一定是 DID 控制器。DID 控制器使用 DID 文档最高级别（数据模型中的顶层映射）的 controller 属性表示；请参阅 5.1.2 DID 控制器。

5.2.1 验证材料

验证材料 是指用于执行 **验证方法** 的任何信息。 **验证方法** 的类型用于确定其与相关过程的兼容性。 **验证材料属性** 的示例包括 `publicKeyJwk` 或 `publicKeyMultibase`。 **加密套件规范** 负责指定 **验证方法** 类型及其对应的验证材料。 例如，请参考 JSON Web Signature 2020 和 Ed25519 Signature 2020。有关 DID 可用的所有注册 **验证方法** 类型及其相关验证材料，请参阅 DID 规范注册表 [DID-SPEC-REGISTRIES]。

为了提高互操作实现的可能性，本规范限制了 DID 文档中表达验证材料的格式数量。 实现者需要实现的格式越少，他们支持所有格式的可能性就越大。 这种方法试图在易于实现和支持历史上广泛部署的格式之间取得微妙的平衡。 下面列出了两种支持的验证材料属性：

- **publicKeyJwk**

  publicKeyJwk 属性不是必选的，如果选择，其值 **必须** 是一个表示符合 [RFC7517] 的 JSON Web Key 的映射。 该映射 **不能** 包含 "d" 或注册模板中描述的私有信息类的任何其他成员。 **建议** 使用 JWKs [RFC7517] 表示其公钥的验证方法使用 `kid` 的值作为其片段标识符。 **建议** 将 JWK `kid` 值设置为公钥指纹 [RFC7638]。 有关具有复合密钥标识符的公钥示例，请参见示例 13 中的第一个密钥。

- **publicKeyMultibase**`

  publicKeyMultibase 属性不是必选的。 此功能是非规范性的。 如果存在，其值 **必须** 是 [MULTIBASE] 编码公钥的字符串表示。

> 请注意，[MULTIBASE] 规范尚未成为标准，可能会发生变化。 在某些情况下，可以使用此数据格式定义 `publicKeyMultibase` 来表达公钥，但未定义 `privateKeyMultibase` 以防止意外泄露私钥。

验证方法不得包含同一材料的多个验证材料属性。 例如，使用 `publicKeyJwk` 和 `publicKeyMultibase` 同时在验证方法中表达密钥材料是不被允许的。

下面显示了一个包含使用上述两种属性的验证方法的 DID 文档示例。

```
例13: 使用 publicKeyJwk 和 publicKeyMultibase 的验证方法
{
  "@context": [
    "https://www.w3.org/ns/did/v1",
    "https://w3id.org/security/suites/jws-2020/v1",
    "https://w3id.org/security/suites/ed25519-2020/v1"
  ],
  "id": "did:example:123456789abcdefghi",
  ...
  "verificationMethod": [{
    "id": "did:example:123#_Qq0UL2Fq651Q0Fjd6TvnYE-faHiOpRlPVQcY_-tA4A",
```

```
    "type": "JsonWebKey2020", // external(property value)
    "controller": "did:example:123",
    "publicKeyJwk": {
      "crv": "Ed25519", // external(property value)
      "x": "VCpo2LMLhn6iWku8MKvSLg2ZAoC-nlOyPVQaO3FxVeQ", // external(property
value)
      "kty": "OKP", // external(property value)
      "kid": "_Qq0UL2Fq651Q0Fjd6TvnYE-faHiOpRlPVQcY_-tA4A" // external(property
value)
    }
  }, {
    "id": "did:example:123456789abcdefghi#keys-1",
    "type": "Ed25519VerificationKey2020", // external(property value)
    "controller": "did:example:pqrstuvwxyz0987654321",
    "publicKeyMultibase": "z6MkmM42vxfqZQsv4ehtTjFFxQ4sQKS2w6WR7emozFAn5cxu"
  }],
  ...
}
```

5.2.2 参考验证方法

验证方法可以嵌入到或引用自与各种验证关系关联的属性中，如 5.3 验证关系中所述。引用验证方法允许它们被多个验证关系使用。

如果验证方法属性的值是一个映射，则该验证方法被嵌入，其属性可以直接访问。然而，如果该值是一个 URL 字符串，则该验证方法是通过引用包含的，其属性需要从 DID 文档的其他位置或另一个 DID 文档中检索。这是通过反引用 URL 并搜索结果资源中的具有匹配 URL 值的 id 属性的验证方法映射来完成的。

例14：嵌入和引用验证方法

```
{
...

  "authentication": [
    // 此密钥被引用，可能被多个验证关系使用
    "did:example:123456789abcdefghi#keys-1",
    // 此密钥被嵌入，仅可用于身份验证
    {
      "id": "did:example:123456789abcdefghi#keys-2",
      "type": "Ed25519VerificationKey2020",
      "controller": "did:example:123456789abcdefghi",
      "publicKeyMultibase": "z6MkmM42vxfqZQsv4ehtTjFFxQ4sQKS2w6WR7emozFAn5cxu"
    }
  ],
```

```
  ...
}
```

5.3 验证关系

验证关系表达了 DID 主题与验证方法之间的关系。

不同的验证关系允许关联的验证方法用于不同目的。 验证者可以通过检查所使用的验证方法是否包含在 DID 文档的相应验证关系属性中来确定验证尝试的有效性。

DID 主题与验证方法之间的验证关系在 DID 文档中是明确的。 不与特定验证关系关联的验证方法不能用于该验证关系。 例如，身份验证属性值中的验证方法不能用于与 DID 主题进行密钥协商协议 - 需要使用密钥协商属性的值。

DID 文档不使用验证关系表达被撤销的密钥。 如果引用的验证方法不在用于反引用的最新 DID 文档中，则该验证方法被视为无效或已撤销。 每种 DID 方法规范都应详细说明如何执行和跟踪撤销。

以下部分定义了几种有用的验证关系。 DID 文档可以包含其中任何一个或其他属性来表达特定的验证关系。 为了最大限度地提高全球互操作性，任何使用的此类属性都应在 DID 规范注册表 [DID-SPEC-REGISTRIES] 中注册。

5.3.1 身份验证

身份验证验证关系用于指定预期如何对 DID 主题进行身份验证，例如登录网站或参与任何类型的挑战-响应协议。

authentication 身份验证属性是可选的。 如果存在，关联值必须是一组一个或多个验证方法。 每个验证方法可以嵌入或引用。

例15: 含三个验证方法的认证属性

```
{
  "@context": [
    "https://www.w3.org/ns/did/v1",
    "https://w3id.org/security/suites/ed25519-2020/v1"
  ],
  "id": "did:example:123456789abcdefghi",
  ...
  "authentication": [
    // 此方法可用于身份验证为 did:...fghi
    "did:example:123456789abcdefghi#keys-1",
```

```
    // 此方法 *仅适用于* 身份验证，不得用于任何其他证明目的，
    // 因此其完整描述在此嵌入，而不是仅使用引用
    {
      "id": "did:example:123456789abcdefghi#keys-2",
      "type": "Ed25519VerificationKey2020",
      "controller": "did:example:123456789abcdefghi",
      "publicKeyMultibase": "z6MkmM42vxfqZQsv4ehtTjFFxQ4sQKS2w6WR7emozFAn5cxu"
    }
  ],
  ...
}
```

一旦身份认证成功，后续操作由 DID 方法或其他应用决定。特定 DID 方法可能认为，以 DID 控制器的身份认证足以更新或删除 DID 文档。但其他 DID 方法可能要求使用不同的密钥或验证方式来执行这些操作。换言之，认证之后的处理不在数据模型的范围内，由 DID 方法和应用自行定义。

这对于任何需要验证认证实体是否持有有效认证证明的认证验证器都很有用。当验证器接收到包含认证证明的数据（特定协议格式）且该数据表明实体由 DID 标识时，验证器会使用 DID 文档中列出的验证方法（例如公钥）来验证该证明。

需要注意的是，DID 文档中的 authentication 属性所指明的验证方法仅用于认证 DID 主题。要认证不同的 DID 控制器，需使用控制器关联实体（如 5.1.2 DID 控制器定义）的 DID 文档和对应的认证验证关系进行认证。

5.3.2 断言

assertionMethod 验证关系用于指定 DID 主题预期如何表达声明，例如用于颁发可验证凭证 [VC-DATA-MODEL]。

- **assertionMethod**
  assertionMethod 属性不是必选的，如果选择，关联值必须是一组一个或多个验证方法。每个验证方法可以嵌入或引用。

例如，在验证者处理可验证凭证期间，此属性很有用。在验证过程中，验证者通过检查用于断言证明的验证方法是否与相应 DID 文档中的 assertionMethod 属性关联，来检查可验证凭证是否包含由 DID 主题创建的证明。

例16：包含两个验证方法的断言方法属性

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

```
  "assertionMethod": [
    // 此方法可以采用与did:...fghi算法类似的方式进行密钥协商
    "did:example:123456789abcdefghi#keys-1",
    // 此方法仅限于密钥协商使用，不会用于任何其他验证关系，因此其完整描述嵌入此处，而非仅使用
引用
    {
      "id": "did:example:123456789abcdefghi#keys-2",
      "type": "Ed25519VerificationKey2020", // external(property value)
      "controller": "did:example:123456789abcdefghi",
      "publicKeyMultibase": "z6MkmM42vxfqZQsv4ehtTjFFxQ4sQKS2w6WR7emozFAn5cxu"
    }
  ],
  ...
}
```

5.3.3 密钥协商

密钥协商验证关系用于指定实体如何生成加密材料，以便传输意图发送给DID主题的机密信息，例如建立与接收方的安全通信通道。

- keyAgreement
  密钥协商属性不是必选的，如果选择，关联值必须是一组一个或多个验证方法。每个验证方法可以嵌入或引用。

该属性的一个使用示例是加密意图发送给DID主题的消息。在这种情况下，对方使用验证方法中的加密公钥信息来包装接收方的解密密钥。

```
Example 17: Key agreement property containing two verification methods
{
  "@context": "https://www.w3.org/ns/did/v1",
  "id": "did:example:123456789abcdefghi",
  ...
  "keyAgreement": [
    // 此方法可以采用与did:...fghi算法类似的方式进行密钥协商
    "did:example:123456789abcdefghi#keys-1",
    // 此方法仅限于密钥协商使用，不会用于任何其他验证关系，因此其完整描述嵌入此处，而非仅使用
引用
    {
      "id": "did:example:123#zC9ByQ8aJs8vrNXyDhPHHNNMSHPcaSgNpjjsBYpMMjsTdS",
      "type": "X25519KeyAgreementKey2019",
      "controller": "did:example:123",
      "publicKeyMultibase": "z6LSn6p3HRxx1ZZk1dT9VwcfTBCYgtNWdzdDMKPZjShLNWG7"
    }
  ],
  ...
```

```
}
```

5.3.4 能力调用

能力调用验证关系用于指定 DID 主题可能用来调用加密能力的验证方法，例如更新 DID 文档的授权。

- **capabilityInvocation**
  能力调用属性不是必选的，如果选择，关联值必须是一组一个或多个验证方法。每个验证方法可以嵌入或引用。

该属性在以下情境中尤为重要：当去中心化身份（DID）主体需访问一个需要授权的受保护HTTP API时。为实现对HTTP API的授权，DID主题使用与特定URL相关联的能力。能力的调用可通过多种形式表达，例如作为一条置于HTTP头中的数字签名消息。

提供 HTTP API 的服务器是能力的验证者，它需要验证被调用的能力所引用的验证方法是否存在于 DID 文档的 capabilityInvocation 属性中。验证者还会检查所执行的操作是否有效，以及能力是否适用于正在访问的资源。如果验证成功，服务器已通过加密方式确定调用者有权访问受保护的资源。

```
Example 18: Capability invocation property containing two verification methods
{
  "@context": [
    "https://www.w3.org/ns/did/v1",
    "https://w3id.org/security/suites/ed25519-2020/v1"
  ],
  "id": "did:example:123456789abcdefghi",
  ...
  "capabilityInvocation": [
    // 此方法可以采用与did:...fghi算法类似的方式进行密钥协商
    "did:example:123456789abcdefghi#keys-1",
    // 此方法仅限于密钥协商使用，不会用于任何其他验证关系，因此其完整描述嵌入此处，而非仅使用引用
    {
    "id": "did:example:123456789abcdefghi#keys-2",
    "type": "Ed25519VerificationKey2020",
    "controller": "did:example:123456789abcdefghi",
    "publicKeyMultibase": "z6MkmM42vxfqZQsv4ehtTjFFxQ4sQKS2w6WR7emozFAn5cxu"
    }
  ],
  ...
}
```

5.3.5 能力委派

`capabilityDelegation` 验证关系用于指定DID主题可能用来将加密能力委派给另一方的机制，例如将访问特定HTTP API的权限委派给下属。

- **capabilityDelegation**

  能力委派属性不是必选的，如果选择，关联值必须是一组一个或多个验证方法。每个验证方法可以嵌入或引用。

一个该属性有用的示例是，当 DID 控制者选择将其访问受保护 HTTP API 的能力委托给其他方时。为了委托这一能力，DID 主题将使用与 `capabilityDelegation` 验证关系相关联的验证方法来对该能力进行加密签名，并将其转交给另一个 DID 主题。受委托方随后将以类似于 5.3.4 能力调用所描述的方式使用该能力。

```
Example 19: Capability Delegation property containing two verification methods
{
  "@context": [
    "https://www.w3.org/ns/did/v1",
    "https://w3id.org/security/suites/ed25519-2020/v1"
  ],
  "id": "did:example:123456789abcdefghi",
  ...
  "capabilityDelegation": [
     // 此方法可以采用与did:...fghi算法类似的方式进行密钥协商
    "did:example:123456789abcdefghi#keys-1",
    // 此方法仅限于密钥协商使用，不会用于任何其他验证关系，因此其完整描述嵌入此处，而非仅使用
引用
    {
    "id": "did:example:123456789abcdefghi#keys-2",
    "type": "Ed25519VerificationKey2020",
    "controller": "did:example:123456789abcdefghi",
    "publicKeyMultibase": "z6MkmM42vxfqZQsv4ehtTjFFxQ4sQKS2w6WR7emozFAn5cxu"
    }
  ],
  ...
}
```

5.4 服务

服务用于 DID 文档中，表达与 DID 主题或关联实体进行通信的方式。服务可以是 DID 主题希望宣传的任何类型的服务，包括用于进一步发现、身份验证、授权或交互的去中心化身份管理服务。

出于隐私考虑，不建议通过服务公开信息，例如社交媒体账户、个人网站和电子邮件地址。关于隐私问题的进一步探讨可以参见 10.1 保持个人数据私密性 和 10.6 服务隐私。与服务相关的信息通常是特定于服务的。例如，与加密消息服务相关的信息可以表达在消息开始之前如何建立加密连接。

服务使用下述 service 属性来表达：

- **service**

  service 属性不是必选的，如果选择，关联的值必须是一组服务，其中每个服务通过一个映射来描述。每个服务映射必须包含 id、type 和 serviceEndpoint 属性。每个服务扩展可以包含附加属性，并且可以进一步限制与扩展相关的属性。

  - **id**

    id 属性的值必须是符合 [RFC3986] 的 URI。符合要求的生产者不得生成具有相同 id 的多个服务条目。符合要求的解析器如果检测到具有相同 id 的多个服务条目，必须生成错误。

    - **type**

      type 属性的值必须是字符串或字符串集。为了最大程度地提高互操作性，服务类型及其关联属性应在 DID 规范注册表 [DID-SPEC-REGISTRIES] 中注册。

    - **serviceEndpoint**

      serviceEndpoint 属性的值必须是字符串、映射，或由一个或多个字符串和/或映射组成的集合。所有字符串值必须是符合 [RFC3986] 的有效 URI，并根据 RFC3986 中的规范化和比较规则及其适用的 URI 方案规范中的任何规范化规则进行规范化。

关于服务的隐私和安全性考虑的更多信息，请参见 10.6 服务隐私、10.1 保持个人数据私密性、10.3 DID 文档相关性风险 和 9.3 身份验证服务端点。

```
Example 20: Usage of the service property
{
  "service": [{
    "id":"did:example:123#linked-domain",
    "type": "LinkedDomains", // external(property value)
    "serviceEndpoint": "https://bar.example.com"
  }]
}
```

6. 表示形式

在本规范中，DID文档的具体序列化形式称为表示形式。表示形式是通过称为生成的过程将数据模型序列化后创建的。表示形式通过称为解析的过程转化为数据模型。生成和解析过程使得信息可以从一种表示形式转换为另一种表示形式。本规范定义了JSON和JSON-LD的表示形式，开发者也可以使用其他能够表达数据模型的表示形式，如XML或YAML。以下部分定义了生成和解析的一般规则，以及JSON和JSON-LD的表示形式。

6.1 生成与解析

除了本规范中定义的表示形式外，实施者可以使用其他表示形式，前提是每种表示形式都被正确指定（包括对未列入DID规范注册表 [DID-SPEC-REGISTRIES]的属性进行互操作处理的规则）。更多信息见4.1 可扩展性。

所有表示形式的要求如下：

1. 表示形式必须为 4. 数据模型 中指定的所有数据类型定义确定性的生成和解析规则。
2. 表示形式必须唯一地与IANA注册的媒体类型相关联。
3. 表示形式必须为其媒体类型定义片段处理规则，并符合片段中定义的片段处理规则。
4. 表示形式应使用数据模型数据类型的词法表示形式。例如，JSON和JSON-LD使用XML Schema的dateTime词法序列化来表示日期时间。表示形式可以选择使用不同的词法序列化来序列化数据模型的数据类型，只要解析过程回到数据模型时是无损的。例如，一些基于CBOR的表示形式使用整数来表示自Unix纪元以来的秒数来表达日期时间值。
5. 表示形式可以定义特定于表示形式的条目，这些条目存储在表示形式特定的条目映射中，以便在生成和解析过程中使用。这些条目用于在解析或生成时，帮助确保无损转换。
6. 为了最大限度地实现互操作性，表示形式规范的作者应在DID规范注册表 [DID-SPEC-REGISTRIES]中注册他们的表示形式。

所有符合规范的生成器的要求如下：

1. 符合规范的生成器必须在生成过程中以DID文档数据模型和表示形式特定的条目映射作为输入。符合规范的生成器可以接受其他选项作为生成过程的输入。
2. 符合规范的生成器必须序列化DID文档数据模型中的所有条目，以及表示形式特定的条目映射中没有明确处理规则的条目，并在生成过程完成后返回序列化结果。
3. 符合规范的生成器必须在生成过程完成后返回与表示形式相关联的媒体类型字符串。
4. 符合规范的生成器不得生成不符合规范的DID或DID文档。

所有符合规范的解析器的要求如下：

1. 符合规范的解析器必须在解析过程中以表示形式和媒体类型字符串作为输入。符合规范的解析器可以接受其他选项作为解析过程的输入。
2. 符合规范的解析器必须使用媒体类型输入字符串来确定DID文档的表示形式。
3. 符合规范的解析器必须检测所有已知表示形式中的任何表示形式特定条目，并将该条目放入表示形式特定的条目映射中，在解析过程完成后返回。所有已知的表示形式特定条目的列表可在DID规范注册表 [DID-SPEC-REGISTRIES]中查阅。
4. 符合规范的解析器必须将所有没有明确处理规则的非表示形式特定条目添加到DID文档数据模型中，并使用仅表示形式的数据类型处理规则，并在解析过程完成后返回DID文档数据模型。
5. 符合规范的解析器在解析不符合规范的DID或DID文档时必须产生错误。

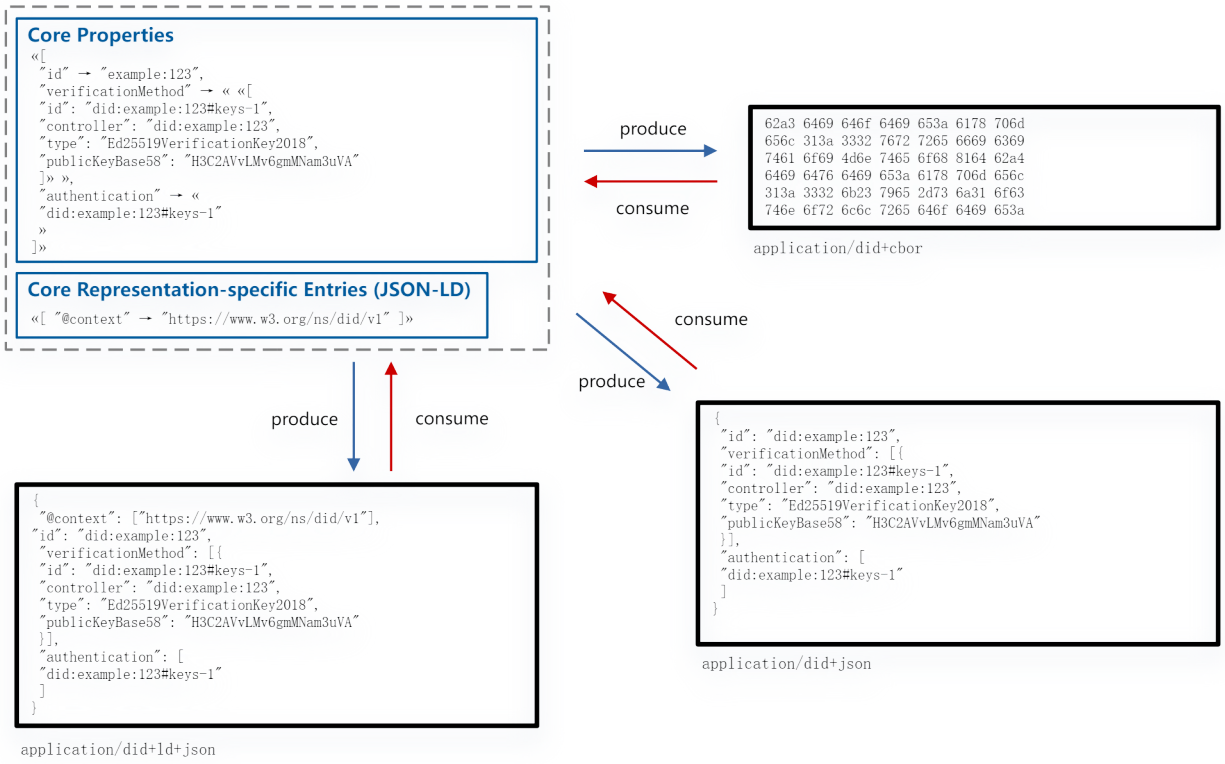

图 4 表示的生成与解析。参见：叙述描述。

注：表示之间的转换 实现应通过在源表示上使用解析规则生成数据模型，然后使用生产规则将数据模型序列化为目标表示，或使用任何其他能生成相同目标表示的机制，在表示之间进行转换。

6.2 JSON

本节定义了 JSON 表示的生成与解析规则。

6.2.1 生成

生产 DID 文档、DID 文档数据结构和特定表示的条目映射必须根据以下生产规则序列化为 JSON 表示：

数据类型	JSON表示类型
map	JSON 对象，其中每个条目作为 JSON 对象的成员序列化，条目的键作为 JSON 字符串成员名，值根据其类型按本表定义进行序列化。
list	JSON 数组，其中列表的每个元素按顺序作为数组的值进行序列化，类型根据本表定义。
set	JSON 数组，其中集合的每个元素按顺序添加为数组的值，类型根据本表定义。
datetime	JSON 字符串，序列化为标准化为 UTC 00:00:00 的 XML 日期时间格式，且不包含秒以下的小数精度。例如：2020-12-20T19:17:47Z`.
string	JSON 字符串。
integer	不带小数或分数部分的 JSON 数字。
double	带有小数和分数部分的 JSON 数字。
boolean	JSON 布尔值。
null	JSON 空值字面量。

> 所有生成 JSON 表示的符合标准的生产者，建议确保其算法与 [INFRA] 规范中的 JSON 序列化规则保持一致，并遵循 JSON [RFC8259] 规范中关于数字精度的建议。

DID 文档的所有条目必须包含在根 JSON 对象中。条目可以包含符合上述值表示规则的额外数据子结构。在序列化 DID 文档时，符合标准的生产者必须为下游应用程序（如 7.1.2 DID 解析元数据中描述的）指定媒体类型为 application/did+json。

```
Example 21: Example DID document in JSON representation
{
  "id": "did:example:123456789abcdefghi",
  "authentication": [{
    "id": "did:example:123456789abcdefghi#keys-1",
    "type": "Ed25519VerificationKey2018",
    "controller": "did:example:123456789abcdefghi",
    "publicKeyBase58": "H3C2AVvLMv6gmMNam3uVAjZpfkcJCwDwnZn6z3wXmqPV"
  }]
}
```

6.2.2 解析

DID 文档和 DID 文档数据结构的 JSON 表示 *必须* 根据以下 解析 规则进行反序列化到 数据模型 中：

JSON 表示类型	数据类型
JSON 对象	一个 映射，JSON 对象的每个成员作为映射中的一个条目添加。每个条目的键设置为 JSON 对象成员名称。每个条目的值通过根据本表中定义的 JSON 表示类型转换 JSON 对象成员值来设置。由于 JSON 对象未指定顺序，因此不保证插入顺序。
JSON 数组，其中 数据模型 条目值为 列表 或未知	一个 列表，JSON 数组的每个值按顺序添加到列表中，根据数组值的 JSON 表示类型进行转换，如本表所定义。
JSON 数组，其中 数据模型 条目值为 集合	一个 集合，JSON 数组的每个值按顺序添加到集合中，根据数组值的 JSON 表示类型进行转换，如本表所定义。
JSON 字符串，其中 数据模型 条目值为 日期时间	一个 日期时间。
JSON 字符串，其中 数据模型 条目值类型为 字符串 或未知	一个 字符串。
JSON 数字，无小数或分数部分	一个 整数。
JSON 数字，具有小数和分数部分，或者当条目值为 双精度浮点数 时，无论是否包含分数部分	一个 双精度浮点数。
JSON 布尔值	一个 布尔值。
JSON null 字面量	一个 null 值。

> 所有创建 符合规范的解析器 并生成 JSON 表示 的实现者建议确保其算法与 [INFRA] 规范中的 JSON 转换规则 以及 JSON [RFC8259] 规范中关于数字的 精度建议 保持一致。

如果符合规范的解析器可获得媒体类型信息，并且媒体类型值为 application/did+json，则正在解析的数据结构是 DID 文档，根元素 *必须* 是一个 JSON 对象，其中对象的所有成员都是 DID 文档 的条目。对于根元素不是 JSON 对象 的 DID 文档，符合规范的解析器 *必须* 报告错误。

6.3 JSON-LD

JSON-LD [JSON-LD11] 是一种基于 JSON 的格式，用于序列化 链接数据。本节定义了 JSON-LD 表示 的 生成 和 解析 规则。

JSON-LD 表示 定义了以下 表示特定条目：

- @context

  JSON-LD 上下文 可以是一个 字符串 或一个 列表，包含任意组合的 字符串 和/或 有序映射。

6.3.1 生成

DID 文档、DID 文档数据结构以及 表示特定条目 映射 *必须* 根据 JSON 表示 的 生成 规则序列化为 JSON-LD 表示，具体定义见 6.2 JSON。

除了使用 JSON 表示 的 生成 规则外，JSON-LD 生成 *必须* 包含 表示特定的 `@context` 条目。 `@context` 的序列化值 *必须* 是 JSON 字符串 `https://www.w3.org/ns/did/v1`，或者是一个 JSON 数组，其中第一个项目为 JSON 字符串 `https://www.w3.org/ns/did/v1`，后续项目根据 JSON 表示 的 生成 规则序列化。

示例 22：有效的简单 @context 条目序列化

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

示例 23：有效的分层 @context 条目序列化

```
{
  "@context": [
    "https://www.w3.org/ns/did/v1",
    "https://did-method-extension.example/v1"
  ],
  ...
}
```

所有创建 符合规范的生产者 并生成 JSON-LD 表示 的实现者建议确保其算法生成有效的 JSON-LD [JSON-LD11] 文档。无效的 JSON-LD 文档将导致 JSON-LD 处理器停止并报告错误。

为了实现不同 表示 之间的互操作性，所有 JSON-LD 上下文及其术语 *应* 在 DID 规范注册表中注册 [DID-SPEC-REGISTRIES]。

生成 JSON-LD 表示 的 符合规范的生产者 *不应* 生成包含未通过 `@context` 定义的术语的 DID 文档，因为符合规范的解析器被期望移除未知术语。在序列化 DID 文档 的 JSON-LD 表示 时，符合规范的生产者 *必须* 为下游应用指定媒体类型 `application/did+ld+json`。

6.3.2 解析

DID 文档 及任何通过 JSON-LD 表示 表达的 DID 文档数据结构 *必须* 根据 JSON 表示 的 解析 规则反序列化为 数据模型，具体定义见 6.2 JSON。

所有创建 符合规范的解析器 并解析 JSON-LD 表示 的实现者建议确保其算法仅接受有效的 JSON-LD [JSON-LD11] 文档。无效的 JSON-LD 文档将导致 JSON-LD 处理器停止并报告错误。

处理 JSON-LD 表示 的 符合规范的解析器 *应* 删除所有未通过 `@context` 定义的 DID 文档 中的术语。

7. 方法

DID 方法定义了实施者如何实现本规范所描述的特性。DID 方法通常与特定的可验证数据注册表相关联。新的 DID 方法在其自身的规范中定义，以便实现不同实现之间的互操作性。

从概念上讲，本规范与 DID 方法规范之间的关系类似于 IETF 通用URI规范[RFC3986]与特定URI方案[IANA-URI-SCHEMES]（如 `http` 方案[RFC7230]）之间的关系。除了定义特定的DID 方案外，DID 方法规范还定义了使用特定类型的可验证数据注册表创建、解析、更新和停用DID和DID文档的机制。它还记录了与DID相关的所有实施考虑因素，以及安全性和隐私考虑。

本节规定了编写DID 方法规范的要求。

7.1 方法语法

所有DID 方法规范在定义方法特定的DID 语法时的要求如下：

1. DID 方法规范*必须*定义一个方法特定的DID 方案，并通过 `method-name` 规则在3.1 DID 语法中指定的一个方法名称进行标识。
2. DID 方法规范*必须*指定如何生成DID的 `method-specific-id` 组件。
3. DID 方法规范*必须*定义 `method-specific-id` 值的敏感性和规范化。
4. `method-specific-id` 值*必须*在DID 方法中是唯一的。 `method-specific-id` 值本身可能是全局唯一的。
5. 任何由DID 方法生成的DID*必须*是全局唯一的。
6. 为了减少 `method-name` 冲突的可能性，DID 方法规范*应*在DID 规范注册处注册[DID-SPEC-REGISTRIES]。
7. DID 方法可以定义多个 `method-specific-id` 格式。
8. `method-specific-id` 格式*可以*包含冒号。冒号的使用*必须*在语法上符合 `method-specific-id` ABNF规则。
9. DID 方法规范*可以*为DID 路径指定比本节中通用规则更严格的ABNF规则。

10. DID 方法规范*可以*为DID 查询指定比本节中通用规则更严格的ABNF规则。
11. DID 方法规范*可以*为DID 片段指定比本节中通用规则更严格的ABNF规则。

注：方法特定的 `method-specific-id` 中的冒号

`method-specific-id` 中冒号的含义完全是方法特定的。DID 方法可能会使用冒号来建立分层的命名空间，用于识别可验证数据注册表中的特定实例或部分，或用于其他目的。建议实施者避免假设与冒号相关的含义或行为，这些含义或行为对所有DID 方法都具有普遍适用性。

7.2 方法操作

所有DID 方法规范在定义方法操作时的要求如下：

1. DID 方法规范 *必须* 定义如何执行所有操作的授权，包括任何必要的加密过程。
2. DID 方法规范 *必须* 指定如何创建DID及其相关的DID 文档。
3. DID 方法规范 *必须* 指定如何使用DID来解析DID 文档，包括DID 解析器如何验证响应的真实性。
4. DID 方法规范 *必须* 指定什么构成对DID 文档的更新，以及DID 控制器如何更新DID 文档*或*声明更新不可能。
5. DID 方法规范 *必须* 指定如何停用DID*或*声明停用不可能。

执行操作的授权方的权威性是特定于DID 方法的。例如，DID 方法可以—

- 使用 `controller` 属性。
- 使用列在 `authentication` 下的验证方法。
- 使用DID 文档中其他构造，例如通过 `capabilityInvocation` 验证关系指定的验证方法。
- 根本不使用DID 文档做出这一决定，而依赖于带外机制。

7.3 安全要求

所有DID 方法规范在撰写*安全考虑*部分时的要求如下：

1. DID 方法规范 *必须* 遵循RFC3552：编写安全考虑部分中提供的所有指导和规范语言，以便于DID操作。
2. 安全考虑部分 *必须* 记录DID操作所面临的以下攻击形式：窃听、重放、消息插入、删除、修改、拒绝服务、放大和中间人攻击。其他已知攻击形式*应该*也进行记录。
3. 安全考虑部分 *必须* 讨论剩余风险，例如在相关协议中的妥协、错误实现或在威胁缓解后使用的密码。
4. 安全考虑部分 *必须* 为7.2 方法操作中要求的所有操作提供完整性保护和更新身份验证。
5. 如果涉及身份验证，特别是用户-主机身份验证，身份验证方法的安全特性 *必须* 清晰记录。
6. 安全考虑部分 *必须* 讨论证明DID唯一分配的政策机制。

7. 方法特定的端点身份验证 *必须* 进行讨论。对于使用具有不同网络拓扑的DLTs的DID 方法，有时作为*轻节点*或*薄客户端*实现，来减少所需的计算资源，必须讨论可供DID 方法实现的拓扑的安全假设。
8. 如果协议包含加密保护机制，DID 方法规范 *必须* 清楚指明哪些数据部分受到保护，以及这些保护的性质，*并且应该*指明加密保护可能面临的攻击类型。例如仅提供完整性保护、保密性和端点身份验证。
9. 应明确标记需要保密的数据（密钥材料、随机种子等）。
10. DID 方法规范*应该*解释并指定对DID 文档的签名实施（如适用）。
11. 在DID 方法使用对等计算资源时，如所有已知的DLTs，应讨论这些资源的预期负担与拒绝服务之间的关系。
12. DID 方法如引入新认证服务类型，应考虑所支持认证协议的安全要求。

7.4 隐私要求

所有DID 方法规范在撰写*隐私考虑*部分时的要求如下：

1. DID 方法规范的隐私考虑部分 *必须* 讨论RFC6973第5节中适用的任何子节，具体考虑的子节包括：监控、存储数据泄露、未请求流量、错误归属、关联、识别、二次使用、披露和排除。

8. 安全考虑

本节为非规范性。

本节包含了一系列安全考虑，建议使用去中心化标识符（DID）的人在将此技术部署到生产环境之前进行考虑。DID旨在在许多IETF标准所使用的威胁模型下运行，且已在[RFC3552]中进行了文档记录。本节详细阐述了[RFC3552]中的多个考虑因素，以及一些DID架构特有的其他考虑因素。

8.1 选择DID解析器

DID规范注册表[DID-SPEC-REGISTRIES]包含了一份DID方法名称及其相应的DID方法规范的信息性列表。实施者需记住，没有中央权威机构规定特定的DID方法规范与任何特定的DID方法名称一起使用。如果对某个特定的DID解析器是否正确实现了DID方法有疑问，可以使用DID规范注册表查找注册的规范，以便做出明智的选择。

8.2 证明控制和绑定

在数字世界或物理世界中将实体绑定到DID、DID文档或加密材料时，需要使用本规范所考虑的安全协议。以下部分描述了一些可能的场景，以及其中的实体如何证明对DID或DID文档的控制，以用于身份验证或授权。

证明对DID和/或DID文档的控制

证明对DID和/或DID文档的控制在更新或与远程系统进行身份验证时非常有用。加密数字签名和可验证时间戳使与DID文档相关的某些安全协议在加密上可验证。为此，本规范在5.3.1 身份验证和5.3.4 能力调用中定义了有用的验证关系。与验证方法关联的秘密加密材料可用于在身份验证或授权安全协议中生成加密数字签名。

注：签名的DID文档

一些DID方法允许在DID文档或元数据结构中包含数字签名和其他证明。然而，这些证明本身并不一定能证明对DID的控制，或保证DID文档是该DID的正确文档。为了获取正确的DID文档并验证对DID的控制，有必要执行根据DID方法定义的DID解析过程。

绑定到物理身份

DID和DID文档本身不携带任何个人数据，强烈建议非公开实体不要在DID文档中发布个人数据。

以一种可以由可信任的权威（如政府）证明的方式，将DID绑定到个人或组织的物理身份可能是有用的。本规范提供了5.3.2 断言的验证关系以实现这一目的。此功能可以促进私密的互动，并在一个或多个管辖区下被视为具有法律约束力；建立这样的绑定必须谨慎平衡隐私考虑（见9. 隐私考虑）。

将DID绑定到物理世界中的某个事物（例如，通过使用与该DID具有相同主题的可验证凭证）的过程在本规范中得到了考虑，并在可验证凭证数据模型[VC-DATA-MODEL]中进一步定义。

8.3 身份验证服务端点

如果DID文档发布了一个服务，旨在验证或授权DID主题（见第5.4节 服务），那么服务端点提供者、主题或请求方有责任遵循该服务端点支持的身份验证协议的要求。

8.4 不可否认性

DID和DID文档更新的不可否认性得到支持，如果：

- 可验证数据注册表支持可验证时间戳。有关可在DID解析过程中使用的有用时间戳的信息，请参见[DID-RESOLUTION]中的"DID文档元数据"。
- 主题正在监控未授权更新，正如在8.5节 DID文档更改通知中详细说明的那样。
- 主题有足够的机会根据DID方法的授权机制撤销恶意更新。

8.5 DID文档更改通知

对DID文档未授权更改的一种缓解措施是监控并主动通知DID主题有更改。这类似于通过向文件上的电子邮件地址发送密码重置通知，帮助防止传统用户名/密码帐户的账户接管。

在DID的情况下，没有中介注册机构或帐户提供商来生成此类通知。然而，如果注册DID的可验证数据注册表直接支持更改通知，则可以为DID控制者提供订阅服务。通知可以直接发送到现有DID中列出的相关服务端点。

如果DID控制者选择依赖于第三方监控服务（而非可验证数据注册表本身），这就引入了另一种攻击向量。

8.6 密钥和签名过期

在去中心化标识符架构中，可能没有中央机构来强制执行加密材料或加密数字签名过期政策。因此，依赖于支持软件（如DID解析器和验证库）来验证请求方在使用时加密材料是否未过期。请求方可能会在其验证过程中应用自己的过期政策。例如，一些请求方可能接受五分钟前的身份验证，而其他具有高精度时间源的请求方可能要求身份验证在过去的500毫秒内进行时间戳。

有些请求方合法地需要延长已过期加密材料的使用，例如验证遗留加密数字签名。在这些情况下，请求方可能会指示其验证软件忽略加密密钥材料的过期，或确定加密密钥材料在使用时是否过期。

8.7 验证方法轮换

轮换是一种管理过程，使现有验证方法关联的秘密加密材料可以在添加新验证方法到DID文档后停用或销毁。此后，任何控制者使用旧的秘密加密材料生成的新证明现在可以改为使用新的加密材料生成，并可以通过新的验证方法进行验证。

轮换是防止验证方法被妥协的有用机制，因为控制者频繁轮换验证方法可以减少单一被妥协验证方法对攻击者的价值。立即在轮换后进行撤销对于控制者指定的短期验证（如加密消息和身份验证）是有用的。

以下考虑可能在考虑使用验证方法轮换时有帮助：

- 验证方法轮换是一项主动安全措施。
- 通常建议定期进行验证方法轮换。
- 高安全环境往往会更频繁地进行验证方法轮换。
- 验证方法轮换只在当前或最新版本的DID文档中表现为变化。
- 当验证方法已使用较长时间，或执行了许多操作时，控制者可能希望执行轮换。
- 验证方法的频繁轮换可能会使被迫持续更新或刷新相关凭证的方感到沮丧。
- 依赖于未在最新版本的DID文档中存在的验证方法的证明或签名不受轮换的影响。在这些情况下，验证软件可能需要额外的信息，例如某个特定验证方法的有效时间，以及访问包含历史记录的可验证数据注册表，以确定证明或签名的有效性。并非所有DID方法都可能提供此选项。
- DID方法操作

人员的能力、以及验证方法本身的复杂性（例如，是否需要具有自我主权能力），会影响轮换的复杂性。

8.8 验证方法撤销

撤销是一个管理过程，使与现有验证方法相关的秘密加密材料被停用，从而不再有效用于创建新的数字签名证明。

撤销是应对验证方法被攻破的有效机制。立即在轮换后执行撤销对控制者指定用于短期验证的验证方法特别有用，例如涉及加密消息和身份验证的验证方法。

与验证方法相关的秘密被攻破后，攻击者可以根据控制者在DID文档中表达的验证关系使用它们，例如用于身份验证。攻击者对这些秘密的使用可能与合法控制者的使用难以区分，从验证方法注册之时起，到撤销之时。

在考虑使用验证方法撤销时，以下几点可能有用：

- 验证方法撤销是一种反应性安全措施。
- 支持密钥撤销被视为最佳实践。
- 控制者应立即撤销已知被攻破的任何验证方法。
- 验证方法撤销只能体现在最新版本的DID文档的更改中；它无法追溯调整以前的版本。
- 如5.2.1 验证材料所述，缺少验证方法是适用于所有支持撤销的DID方法的唯一撤销形式。
- 如果验证方法不再仅限于控制者或被信任的代表控制者的各方访问，预计应立即撤销以降低冒充、盗窃和欺诈等风险。
- 撤销应被理解为控制者表示与撤销的验证方法相关的证明或签名在撤销后应视为无效。这也可能暗示存在担忧，即现有的证明或签名可能由攻击者创建，但这并不一定是事实。然而，验证者可能仍然选择自行决定接受或拒绝任何此类证明或签名。
- 关于DID方法操作的部分规定了DID操作，支持DID方法规范，包括更新和停用，可以用于从DID文档中删除验证方法。
- 不是所有的DID方法都支持验证方法撤销。
- 即使验证方法出现在DID文档中，额外信息，如公钥撤销证书或外部允许或拒绝列表，可能用于确定验证方法是否已被撤销。
- 依赖于被攻破的验证方法的任何软件的日常操作，例如个人的操作系统、杀毒软件或终端保护软件，可能会受到影响，当验证方法被公开撤销时。

撤销语义

尽管验证者可能选择不接受来自撤销的验证方法的证明或签名，但判断验证是否是通过撤销的验证方法进行的并不像看起来那么简单。一些DID方法提供回溯到DID在某一时间点的状态，或DID文档的特定版本的能力。当这样的特性与可靠的方法结合以确定在加密可验证声明做出时的时间或DID版本时，撤销并不撤销该声明。这可以作为使用DIDs进行约束承诺的基础；例如，签署抵押贷款。

如果满足这些条件，撤销不是追溯性的；它仅使该方法的未来使用失效。

然而，为了使这些语义安全，第二个条件——能够知道在断言做出时DID文档的状态——应适用。没有这一保证，某人可能会发现一个被撤销的密钥并使用它在过去的模拟日期做出加密可验证的声明。

一些DID方法仅允许检索DID的当前状态。当这种情况发生时，或者当在某个时间点无法可靠地确定DID在加密可验证声明时的状态时，唯一安全的做法是禁止任何考虑与时间相关的DID状态，除了现在这一时刻。DID生态系统采取这种方法，本质上将加密可验证的声明视为可以在任何时候被DID控制者作废的短暂令牌。

在无信任系统中的撤销

无信任系统是指所有信任源自于可加密证明的声明的系统，尤其是没有其他元数据被纳入到信任确定中。为了验证在无信任系统中已撤销的验证方法的签名证明，DID方法需要支持 `versionId` 或 `versionTime` ，以及DID文档元数据属性 `updated` 和 `nextUpdate` 。验证者只有在以下所有情况都为真时，才能验证已撤销密钥的签名或证明：

- 证明或签名包含在创建签名或证明时使用的DID文档的 `versionId` 或 `versionTime` 。
- 验证者能够确定签名或证明的制作时间点；例如，它是锚定在区块链上的。
- 对于解析的DID文档元数据， `updated` 时间戳在签名或证明制作的时间点之前， `nextUpdate` 时间戳在之后。

在愿意在没有此保证的情况下处理撤销的场

景中，可能会导致错误地接受或拒绝签名或证明。

8.9 DID恢复

恢复是一种反应性安全措施，使失去执行DID操作能力的控制者（例如因设备丢失）能够重新获得执行DID操作的能力。

在考虑使用DID恢复时，以下几点可能有用：

- 定期以不频繁的方式主动进行恢复，有助于确保未失去控制。
- 被视为最佳实践，切勿将与恢复相关的加密材料用于其他任何目的。
- 恢复通常与验证方法轮换和验证方法撤销同时进行。
- 当控制者或信任的服务无法独占执行DID操作时，建议进行恢复，如7.2 方法操作所述。
- DID方法规范可能选择支持可信方的法定人数来促进恢复。有关如何实现的建议见5.1.2 DID控制器。
- 并非所有DID方法规范都会承认使用其他DID方法注册的DIDs的控制，并且可能将第三方控制限制为使用相同方法的DIDs。
- DID方法规范中的访问控制和恢复也可以包括时间锁定功能，以防止通过保持恢复的控制第二轨道来保护密钥被攻破。
- 当前没有适用于所有DID方法的常见恢复机制。

8.10 人类友好标识符的角色

DID 实现全球唯一性而无需中央注册机构，但这会牺牲人类可记忆性。能够生成全球唯一标识符的算法生成的是没有人类意义的随机字符字符串。这种权衡通常被称为 Zooko's Triangle。

在某些情况下，从人类友好标识符出发发现 DID 是可取的。例如，自然语言名称、域名或传统地址（如移动电话号码、电子邮件地址、社交媒体用户名或博客 URL）都可以作为 DID 控制者 的标识符。然而，将人类友好标识符映射到 DID 的问题，以及以可验证和可信的方式进行映射的问题，超出了本规范的范围。

针对这一问题的解决方案在其他规范中定义，例如 [DNS-DID]，这些规范引用了本规范。强烈建议此类规范仔细考虑以下内容：

- 许多基于误导用户关于目标实体真实人类友好标识符的安全攻击。
- 使用人类友好标识符所带来的隐私后果，特别是在它们本质上是可关联的，尤其是如果它们是全球唯一的。

8.11 DIDs 作为增强的 URN

如果 DID 控制者 希望，DID 或 DID URL 可以作为持久、位置独立的资源标识符。这类标识符被归类为统一资源名称（URN），并在 [RFC8141] 中定义。DID 是一种增强的 URN，提供加密安全的、位置独立的数字资源标识符，同时提供可用于检索的元数据。由于 DID 文档 和 DID 之间的间接关系，DID 控制者 可以在不调整 DID 的情况下调整资源的实际位置，甚至直接提供资源。此类 DID 可以确认证明所检索的资源确实是被识别的资源。

意图将 DID 用于此目的的 DID 控制者 被建议遵循 [RFC8141] 中的安全考虑。特别是：

- DID 控制者 应选择一种满足其持久性要求的 DID 方法。去中心化特性评估表 [DID-RUBRIC] 是帮助实施者选择最合适的 DID 方法 的工具之一。
- DID 控制者 应公开其操作政策，以便请求方可以确定他们在多大程度上可以依赖于该 DID 的持久性。在没有此类政策的情况下，请求方不应假设某个 DID 是同一 DID 主题 的持久标识符。

8.12 不可变性

许多网络安全滥用行为利用现实与理性、善意参与者的假设之间的差距。DID 文档的不可变性可以提供一些安全利益。各个 DID 方法 应考虑采取限制措施，以消除不必要的行为或语义。一个 DID 方法 越是“锁定”，同时提供相同的功能，就越不容易被恶意参与者操控。

例如，考虑对 DID 文档 的一次编辑可以改变文档中除根 `id` 属性外的任何内容。但是，服务在定义后是否真的希望更改其 `type`？或者密钥更改其值？或者在某些基本属性发生变化时，要求使用新的 `id` 是否更好？恶意接管网站的攻击往往旨在实现一个结果，即网站保持其主机名称标识符，但在内部却悄然发生变化。如果网站某些属性，例如与其 IP 地址相关联的 ASN 被要求遵循不可变性规范，那么异常检测将变得更容易，攻击将更加困难且代价更高。

对于与全球真实来源相关的 DID 方法，始终可以直接、即时地查找最新版本的 DID 文档。然而，似乎最终会有多层缓存位于 DID 解析器 和真实来源之间。如果是这样，相信 DID 文档 中某个对象的属性具有某种状态，而它们实际上微妙地不同，这可能会引发利用。这一点在某些查找完整 DID 文档 的情况下尤为真实，而其他查找仅涉及部分数据，假设更大的上下文。

8.13 DID 文档中的加密数据

加密算法因加密学和计算能力的进步而可能失败。实施者应假设，放置在 DID 文档 中的任何加密数据最终可能会向与加密数据相同的受众以明文形式提供。这在 DID 文档 是公共的情况下尤其重要。

加密整个或部分 DID 文档 不是保护数据的适当手段。同样，将加密数据放入 DID 文档 也不是保护个人数据的适当方式。

考虑到上述警告，如果在 DID 文档 中包含加密数据，建议实施者不要关联任何可以用来推断加密数据与相关方之间关系的可关联信息。可关联信息的例子包括接收方的公钥、已知在接收方控制下的数字资产的标识符或接收方的可读描述。

8.14 等价性属性

由于 `equivalentId` 和 `canonicalId` 属性是由 DID 方法 本身生成的，因此适用于 DID 文档 中 `id` 字段的相同安全性和准确性保证也适用于这些属性。 `alsoKnownAs` 属性不保证是等价性陈述的准确表达，且不应在未进行超出 DID 文档 解析的验证步骤的情况下依赖。

`equivalentId` 和 `canonicalId` 属性表达对同一 DID 变体的等价性声明，这些变体是由相同的 DID 方法 生成的，可以被信任，前提是请求方信任 DID 方法 和符合标准的生成者及解析者。

`alsoKnownAs` 属性允许对不受同一 DID 方法 管辖的 URI 进行等价性声明，但在未进行超出治理 DID 方法 的验证步骤之前，不能被信任。请参见 5.1.3 也称为 的额外指导。

与 DID 文档 中的任何其他安全相关属性一样，依赖于 DID 文档 中任何等价性声明的各方应防范这些属性的值在进行适当验证后被攻击者替换。任何对存储在内存或磁盘上的 DID 文档 的写入访问都是一种攻击向量，可能会绕过验证，除非对 DID 文档 进行重新验证。

8.15 内容完整性保护

包含指向外部机器可读内容（如图像、网页或模式）的链接的 DID 文档 易受篡改。强烈建议使用哈希链接 [HASHLINK] 等解决方案保护外部链接的完整性。如果外部链接无法得到完整性保护，而 DID 文档 的完整性又依赖于该外部链接，则应避免使用外部链接。

一个外部链接的例子是 JSON-LD 上下文 [JSON-LD11]，其可能影响 DID 文档 本身的完整性。为了防止被攻击，建议 DID 文档 的使用者缓存 JSON-LD 上下文的本地静态副本和/或验证外部上下文的完整性，以确保与安全版本的外部 JSON-LD 上下文关联的密码学哈希相匹配。

8.16 持久性

DID 的设计目的是持久的，这样 控制者 不必依赖单一的受信任第三方或管理员来维护其标识符。在理想情况下，没有管理员可以从 控制者 手中夺走控制权，管理员也不能阻止其标识符用于任何特定目的，例如身份验证、授权和证明。任何第三方都无法代表 控制者 删除或使实体的标识符失效，除非得到 控制者 的同意。

然而，重要的是要注意，在所有启用密码学控制证明的 DID 方法 中，证明控制权的手段始终可以通过转移秘密密码材料转移给另一方。因此，依赖于标识符持久性的系统需要定期检查，以确保该标识符实际上仍然在预定方的控制之下。

不幸的是，单凭密码学无法确定与给定 验证方法 关联的秘密密码材料是否被泄露。预期的 控制者 可能仍然能够访问秘密密码材料，因此可以在验证过程中执行控制证明，同时，一个恶意行为者也可能访问这些相同的密钥或其副本。

因此，密码学控制证明预期仅作为评估高风险场景所需身份保证级别的一个因素。基于 DID 的身份验证提供了比用户名和密码更大的保证，得益于能够在不在系统之间传输该秘密的情况下确定对密码秘密的控制。然而，这并不是万无一失的。涉及敏感、高价值或生命关键操作的场景预计将根据需要使用其他因素。

除了不同 控制者 使用可能带来的潜在歧义外，通常也无法保证在任何给定时刻某个特定 DID 正在被用于指代同一主题。从技术上讲，控制者可以将某个 DID 重新用于不同的主题，更微妙的是，主题的确切定义可能随时间变化或被误解。

例如，考虑一个用于个体经营的 DID，接收用于金融交易的各种凭证。对 控制者 来说，该标识符指代业务。随着业务的发展，最终以有限责任公司形式注册。 控制者 继续使用同一个 DID，因为对他们而言，该 DID 指的是该业务。然而，对国家、税务机关和地方政府而言，该 DID 不再指代同一实体。信贷提供商或供应商是否在意这种微妙的意义变化，必然取决于他们自己。在许多情况下，只要账单能按时支付，催款就能得到执行，这种变化就是无关紧要的。

由于这些潜在的歧义，DID 应被视为有效的 *上下文性* 而非绝对的。它们的持久性并不意味着它们指代确切相同的主题，也不意味着它们在同一 控制者 的控制之下。相反，人们需要理解 DID 创建时的上下文、它的使用方式，并考虑其含义的可能变化，同时采取程序和政策来应对潜在和不可避免的语义漂移。

8.17 保障级别

关于身份验证事件安全上下文的附加信息通常因合规原因而需要，尤其是在金融和公共部门等受监管领域。这些信息通常被称为保障级别（LOA）。示例包括对秘密密码材料的保护、身份验证过程以及身份验证器的形态。

支付服务（PSD 2） 和 eIDAS 引入了对安全上下文的要求。保障级别框架由法规和标准（如 eIDAS、NIST 800-63-3 和 ISO/IEC 29115:2013）分类和定义，包括其对安全上下文的要求，并对如何实现这些要求提出建议。这可能包括强用户身份验证，其中 FIDO2/WebAuthn 可以满足该要求。

某些受监管的场景要求实施特定的保障级别。由于在这些情况下，诸如 `assertionMethod` 和 `authentication` 的 验证关系 可能会被使用，因此有关所应用安全上下文的信息可能需要向 *验证者* 表达和提供。在 DID 文档 数据模型中是否以及如何编码这些信息超出了本规范的范围。有兴趣的读者可以注意到：1）该信息可以使用可验证凭证 [VC-DATA-MODEL] 进行传输，2）可以扩展 DID 文档 数据模型以包含此信息，如 4.1 可扩展性 中所述，并且 9. 隐私考虑 适用于此类扩展。

8.18 评估竞争考虑

本节为非规范性。

本规范不要求或建议使用任何特定类型的 可验证数据注册处。不同的用例可能会产生不同的要求。不同的要求可能会提出不同的考虑和权衡。例如，在计算（能源使用）、信任（对权威的依赖）、协调（网络带宽）或内存（物理存储）之间的权衡可能适用于任何给定用例，也可能不适用。其他用例可能不会进行相同的权衡。需要考虑不同标准的用例被引导至 DID 方法标准，该标准提供评估标准，以帮助决策者确定特定 DID 方法 是否适合其用例。