



> > Azure Active Directory Identity  
> Home Directory Blog  
> > Identity Hubs as personal datastores

[Back to Blog](#)[< Newer Article](#)[Older Article >](#)

Alex Simons (AZURE) Microsoft

03-29-2019 01



## Identity Hubs as personal datastores

Today's world of concentrated social media, data breaches, election hacking, and Cambridge Analytica makes it self-evident that individuals need a new way forward, one which enables them to take back control of all aspects of their identity; one which spans both the physical and digital world.

Being bystanders in the data economy is no longer tenable. Each of us needs to be the master of our own data. We need a way to do this that doesn't assume that service providers will honor our wishes. We need a secure, encrypted mechanism for data storage and exchange that puts each of us in control of our data.

Presently, most of our digital identity and personal data is controlled by a few central service providers. These providers, generally corporations, control our data, including having the ability to deny or revoke access to it.

At Microsoft we believe that the world needs a new model. Our vision for this model is simple:

***Each of us needs a digital identity we own, one which securely and privately stores all elements of our digital identity. This self-owned identity must be easy to use and give us complete control over how our identity data is accessed and used.***

We believe that Decentralized Identities rooted in distributed ledger technologies (blockchains and other related technologies) are the best option for achieving this vision. Decentralized identities give users complete control of their personal data, and how it is accessed. They remove the ability of

central entities (including Microsoft) to stand between a user and their apps, services, and relationships.



To learn more about our vision, you can visit [Decentralized identity](#). Today, we are announcing the next step in our journey to turn this vision into reality.

**With this blog post we are announcing our intent to contribute code to the open source reference implementation of Decentralized Identity Foundation (DIF) Identity Hub.**

## What is an Identity Hub?

**NOTE:** If you aren't familiar with the general architecture of decentralized identities proposed by the DIF, you should start by reading the overview of the system and approach in our [white paper](#).

In a decentralized identity system, identifiers and public keys can be anchored to a variety of Distributed Ledger Technologies (DLT), such as, Bitcoin, Ethereum, and a variety of other technologies that comply with standards being defined by the community via [Decentralized Identity Foundation \(DIF\)](#) and W3C Credentials Community Group.

These ledgers are well suitable for providing the immutable foundation for [Decentralized Identifiers](#), but should not be used to store personal identity data. This would be at odds with the goals of privacy-by-design and with existing and emerging data regulations, such as the GDPR.

Instead, we need a different solution for secure storage of personal data and information.

Identity Hubs are that solution. Identity Hubs are decentralized, off-chain, personal datastores that put control over personal data in the hands of users. They allow users to store their sensitive data—identity information, official documents, app data, etc.—in a way that prevents anyone from using their data without their explicit permission.

Users can use their Identity Hubs to securely share their data with other people, apps, and businesses, providing access to the minimum amount of data necessary, while retaining a record of its use.

App developers can reduce the complexity of data management and compliance by storing sensitive data in a user's Hub. This approach reduces the app developers' risk of privacy violations and data breaches since the data no longer needs to be stored by the application.

The Identity Hub spec is being developed in DIF along with an open source reference implementation. This helps developers and providers who may want to run a Hub on their own devices or infrastructure better understand the APIs and architecture.

The source code for the Identity Hub packages can be found in [DIF's GitHub repositories](#). When development of the DIF Identity Hub reference implementation is complete, users will be able to run their own Hub wherever they want, retaining full control over where their data is stored and managed.

A key property of the Hub is the ability to sync and replicate across device(s) and Cloud(s). This means that users can run their Identity Hubs anywhere, with any provider they choose, for instance on Microsoft infrastructure or anywhere else. Such Identity Hubs can sync data across instances to ensure there is no single point of vulnerability. None of the DIF compliant providers would have means to see who is sharing what data and with whom.

The rest of this blog describes the key features of Identity Hubs, and how they enable stronger user control and privacy over their data.

## Interfaces

The proposed draft spec for Identity Hubs being considered by DIF community describes three key interfaces:

Interface	Description
Collections	The collections interface is the primary interface for storage of personal data. Any semantic data that pertains to the user should be stored using this interface.
Profile	A special object describing basic, public information about a DID, which can be accessed using the Profile interface. This may include the type of DID (user, business, device, etc.), general descriptions, and contact information, and other typical profile details. <i>Open item: we are working on adding a purposeful sample code to demonstrate intended use</i>
Permissions	The Permissions interface registers, manages, and enforces hybrid object capabilities, which are used to create access and encryption boundaries within Hub instances.

When programming against Identity Hubs, choose the interface that is most appropriate for your scenario. In most cases, this will be the Collections interface. There are several other interfaces described in the DIF Identity Hub specification that are still under active development. We plan to implement them all as these parts gain further consensus and our resources allow.

## Semantic data model



As opposed to traditional datastores an Identity Hub does not require predefined schemas or database models. Instead, Hubs follow an open, dynamic, [semantic data model](#) that allows users and the individuals, apps, and services they authorize to store and interact with any data represented via a semantic JSON schema.

As a simple example, a music playlist stored in a Hub could be represented as:

```
{
  "@context": "http://schema.org",
  "@type": "MusicPlaylist",
  "name": "Classic Rock Playlist",
  "numTracks": "5",
  "track": [
    {
      "byArtist": "Lynyrd Skynyrd",
      "duration": "PT4M45S",
      "inAlbum": "Second Helping",
      "name": "Sweet Home Alabama",
    }
  ]
}
```

In this example, the @context and @type metadata fields inform the Hub that the object is a music playlist. When written to the Hub using the Collections interface, it will store the playlist alongside other music playlists of the same type. Different applications can be permissioned by the user to read and write music playlists in the user's Hub, where they can all work from a common set of music playlist entries. App developers can leverage existing industry-specific schemas, such as those described at [schema.org](#), [gs1.org](#), and [hl7.org](#). If desired, any user or developer can publish a custom schema that apps and services can use to collaboratively store and access data.

Using an existing, widely accepted schema for storing an app's data is not required, but highly recommended. To do so, developers need to make sure to set the correct context/type fields and data properties on objects they wish to interact with, and the Hub will take care of the rest. This open semantic data model ensures Hubs can act as generic personal data stores that can flex to accommodate a wide range of data and use cases—including personal info, app data, medical records, etc.

Currently, our version of Identity Hubs only supports simple JSON data. Images, videos, byte streams, raw files, and other content types are not supported at this time. The Hub service API tutorial describes how JSON data is sent to and from the cloud Identity Hub service.



## Commits

Each individual piece of data in an Identity Hub is called an "object." And each object in an Identity Hub is comprised of a series of "commits."

A commit in the context of an Identity Hub is like a git commit to a repo, in that it represents a change to an object. To write data to an Identity Hub, we need to construct and send a new commit to the Hub.

To read data from an Identity Hub, it is necessary to fetch all relevant commits from the Hub and construct an object's current value by applying all its commits in order. The use of commits to represent data in Identity Hubs offers a few distinct advantages:

- Enables Hubs to communicate over a conflict-free replication protocol, enabling multiple Hub instances to sync data between them.
- It creates an auditable history of changes for an object, especially when each commit is signed by a DID.
- It eases implementation for app and service use cases that require offline data modification, where multi-writer conflicts can arise.


The exact details of commits, including how to currently read and write them to Identity Hubs, are included in our Hub service API reference and Hub tutorials.

Clearly typical app developers should not have to know about the intricacies of commit processing: caching and watermarked deltas should provide optimization and simplification of the programming interface so a simple write or read is normally adequate. However, our current version of Identity Hubs does not include such refinements. Yet it is instructive and fully usable by developers who want to evaluate a preview version.

## Permissions

**NOTE:** Identity Hub Permissions will not be easy for end users to interact with until User Agent wallets create graphical interfaces to support their creation and management. For the time being, developers can leverage sample code to understand the key Permissions concepts.

The Identity Hub's permission system ensures that users retain full, fine-grained control over access to their data. As long as the user has sole access to their DID's private key(s), nobody else can gain access to data stored within the Identity Hub.

Access to data in Identity Hubs is controlled through the Permissions interface  using PermissionGrant objects. By rule, the DID that owns the Identity Hub always has full access to read, modify, write, and delete any and all data in an Identity Hub. DID established as owner for an Identity Hub can create PermissionGrant objects in their Hub to grant other DIDs to access data. And because PermissionGrants are modified using commits, users retain an auditable log of who had access to what data at what times. For instance:

```
{
  "@context": "https://identity.foundation",
  "@type": "PermissionGrant",
  "owner": "did:example:abc123",
  "grantee": "did:example:xyz456", // who is being granted access
  "context": "https://schema.org",
  "type": "MusicPlaylist", // what objects are being accessed
  "allow": "-R--", // allows create, read, update, and delete
}
```

PermissionGrants are modeled just like any other object in the Hub and can be accessed through the Permissions interface. When this grant is created in an Identity Hub, it gives the DID did:example:xyz456 the permission to read all music playlists in the user's Identity Hub.

Permission grants have additional controls to restrict data access at a more granular level. For full details on the Hub's permission system, refer to our Hub service API reference.

You can see the full reference for the Permissions spec available [here](#).

## Encryption

**NOTE:** The ability to interact with our Identity Hub instance is meant for developer exploration only – the encryption section of the specification and implementation is not yet implemented. Please do not store any personal and/or sensitive information at this time.

The final aspect of Identity Hubs that ensures data protection is the Hub's encryption subsystem. The goal of the Hub's encryption system is to encrypt all data at the edge, before the data is ever passed to a Hub instance. This way, the Hub provider never sees an object's contents in clear text.

Currently, data encryption support for the Hub is still in development. In the meantime, our Identity Hub cloud service protects data in two other ways:

- Data sent on the wire to a Hub is encrypted with the DID keys associated with the target Hub. This ensures data is confidential in transit.

- Data stored in our Hub service is encrypted using traditional database encryption techniques. This helps ensure data is secure at rest.

## Replication

**NOTE:** Identity Hub replication is still undergoing specification in DIF, and we are actively prototyping the feature as of this writing.

Another aspect of Identity Hub is that a single DID can have many Hub instances. All Hub instances replicate data to one another, so that the user's personal data store is distributed across several different physical data stores.

An Identity Hub can run in the cloud, on edge devices, or any infrastructure that can implement the Identity Hub's replication protocol. This means that users do not have to rely on a single cloud provider, like Microsoft, to act as a single, centralized custodian for their personal data. Instead, users can maintain a replicated instance of their Identity Hub on devices they physically own. They can also set up multiple cloud Hub providers, to reduce risk that their data will become unavailable. In addition, we are also working on and contributing to the community with additional connectors and stores that will allow implementation choice such as local data stores.

## Privacy

**NOTE:** The ability to interact with our Identity Hub instance is meant for developer exploration only – many sections of the specification and implementation are undergoing active development. Please do not store any personal and/or sensitive information at this time.

It's clear that users need to maintain strict separation between different contexts in their lives, employing pairwise DIDs for each. At the same time, a user's Identity Hub needs to be easily discoverable so that apps and services can locate and communicate with the datastore. The Identity Hub design must ensure that two DIDs cannot be correlated based on Identity Hub metadata used for commination.

Identity Hubs maintain privacy in three ways. First, a pointer to a DID's Identity Hub is included in each DID document. This makes the Hub discoverable. Second, Identity Hub providers like Microsoft use a single URL for receiving requests (to eliminate DID target exposure over the wire), and route requests to the appropriate Hub instance behind the scenes. Third, all data and metadata in a Hub request is encapsulated in a JWE that is encrypted with the DID keys of the Hub provider. This makes it impossible for an eavesdropper to infer which DID a Hub request pertains to. Users can create as many DIDs as needed and be confident their Hubs won't divulge any relationship between them.

## Microsoft's Identity Hub Instance

At Microsoft we are both contributing to the open source reference implementation of DIF's Identity Hub project and developing our own instance, optimized for large-scale. When Microsoft's instance is fully operational, it will provide a secure, geo-redundant Identity Hub service that is backed by Azure's operational guarantees. The Identity Hub service follows our [Security Development Lifecycle guidelines \(SDL\)](#) and is deployed in accordance with our safe deployment practices.

## Summary of capabilities for early version of Identity Hub version

Available now:

1. Microsoft cloud hosted Identity Hub instance (early dev version).
2. Commit based JSON object storage and retrieval using a semantic data model.
3. Request authentication and encryption on the wire to ensure privacy.
4. Simple permission model to grant third-parties' access to data.
5. NodeJS SDK for reading and writing to Identity Hubs.

Up next:

1. Hub implementations that run on edge devices locally.
2. Edge data encryption to ensure Hubs don't have access to raw data.
3. Replication protocol to enable multiple Hub instances.
4. Addition of new commit strategies, such as CRDTs and last-write wins.
5. Rich queries for fetching Hub data.

... and much more!

We are working within the DIF's Storage and Compute Working Group on a quick start guide for running an Identity Hub for development purposes. We hope early exposure to this work sparks the imagination to create new apps and services that place users at the center, while maintaining privacy and security.

We always love to hear your feedback and suggestions, and we look forward to hearing from you! Let us know what you think in the comments below.

Best regards,


Alex Simons (@[Alex A Simons](#) )

Corporate VP of Program Management

Microsoft Identity Division

Tags:   AAD     AzureAD     digital identity     Privacy



 11 Likes


## 7 Comments



**Craig Debbo** Contributor

03-29-2019 09:2

Wow, so then, for example, Facebook could support this and store my likes, photos, status-updates in my private ID storage instead of on their servers. Similar to Solid project that Tim Berners-Lee is working on. Maybe this is an implementation of that? Huge economic ramifications if many users do this.


 1 Like



**Alex Simons (AZURE)** Microsoft

03-29-2019 12:0

Hi Craig - yep, that's exactly the idea. We believe it has the potential to create an entirely new model for trustworthy identification that will have far reaching implications and will be a huge win in terms of giving users back their privacy. Regards, Alex


 3 Likes



**Raynor\_Mallory** Occasional Visitor

03-29-2019 02:2


Incredibly important and forward thinking initiative. I'm impressed that Microsoft is investing in this. Very pro-consumer, pro-user. I share your vision. Excited to see this roll out.


 1 Like



**Andre van den Berg** MVP

03-30-2019 08:3

Very interesting to see how it go futher and when it's rolling out in preview to use it. 


 0 Likes



**Claus Kjøller Skovholm Hansen** Senior Member

03-30-2019 11:5

Global ID. Sound wild futuristick (but will for sure happen on day).  
How much CPU power would this (block chain) secure ID mechanism require?  
Bitcoins mining is already causing heavy power consumption. And the mathmatical  
model used must be very secure, else there can be hacking on a global scale.  
Anyway! Very intereresting project


 0 Likes



**agroppe** Occasional Visitor

04-01-2019 06:1

The overwhelming standard for protecting personal data APIs is OAuth2. The  
"permissions" API standard on top of OAuth2 is Kantara UMA. The UMA Authorization  
Server would be an ideal service for the Identity Hub. It would be a service endpoint in a  
DID document and would benefit greatly from the privacy features of the Microsoft  
design. Can the Azure Identity Hub support an UMA Authorization Server? If not, why  
not?


 0 Likes



**beaule** Occasional Visitor

05-19-2019 12:5

Nice !  
How do you position it next to Solid from MIT?

 0 Likes



You must be a registered user to add a comment. If you've already registered, sign in. Otherwise, register and sign in.

[Comment](#)

What's New ▾


Microsoft Store ▾

Education ▾

Enterprise ▾

Developer ▾

Company ▾

 English (United States)