Refining the Design Principles Identity Relationship Management

Abstract:

# Introduction

Identity management is not a vacuum; people and connected devices rarely, if ever, act on their completely on their own devoid of any organizational, personal, and situational context. This has been known to identity management practitioners for many years. This is why identity management progressed from managing individual digital identities to managing groups to managing roles. Each progression gave identity management practitioners great ability to manage larger populations of users. However, the industry’s last progression to role management does not provide enough managerial leverage to adequately tackle such issues as self-sovereign identity, connected devices, and the Internet of Things. Furthermore, in order to more accurately portray the richness of the use cases facing the identity management industry, something more is needed beyond managing identities living in the vacuum of a directory.

What is needed are relationships.

## Purpose and Audience

In 2014, the identity management industry began to discuss the notion of Identity Relationship Management (IRM) and how relationships could provide the richness needed to represent our hyper-connected world and give administrative leverage to identity management professionals. The Kantara Initiative formed the IRM Working Group which in turn produced its “Laws of Relationship Management” report in early 2015. Subsequently, the IRM WG examined the original design principles of IRM and sought out examples of IRM currently implemented. This document is the summation of that effort and is intended for people designing complex identity systems and interactions. This report will not offer prescriptive design patterns for large-scale relationship-oriented systems, but instead offers design principles for consideration and real-world use cases for further study.

# Refined Design Principles of Relationships

Relationships are not new but their representation has rarely been first class citizen in the realm of digital identity. In the past, relationships have been represented as attributes such as memberOf or implied in identifiers such as distinguished name. IRM seeks to promote relationships, in both representations and awareness, in order to provide identity management practitioners with more accurate, more manageable, and more deployable means of managing digital identities in our hyper-connected world. To that end, the IRM WG offers the following design principles of relationships.

## Provable

The existence of a relationship between actors ( be those individuals, groups, organizations, non-human entities, or any combination of these) carries meaning and provides large context. As such, systems handling relationships need a way to state authoritatively that a given relationship exists.

Care must be taken to ensure that the existence of a relationship is only provided to the right parties for the right purposes. For example, if a previously unknown actor asks a doctor’s office, “Is Alice a patient here?” and in doing so is asking for prove that a relationship exists between Alice and the doctor’s office, then the doctor’s office should vet the unknown actor before providing an answer. It may be that the unknown actor is a hospital system in another country where Alice is traveling and she requires medical assistance, and the actor’s request is appropriate and valid. It may be that the unknown actor is a gossip columnist and is looking for salacious morsels to report about Alice.

Keep in mind that in some cases, the actor asking for the existence of the relationship may be one of the parties in the relationship. For example, Bob could ask his employer for the existence of their relationship so Bob can present it to a bank to get a loan. Similarly, Eve may ask a credit bureau for proof of a relationship as her first step to correcting inaccurate information.

Responding to a request of relationship existence requires that the party making the request and the purpose of the request are appropriate as well as that members of the relationship have either explicitly or implicitly agreed that other parties can receive proof of relationship information.

There are several implications of the Provable design principle:

* Proof of Relationship information can be sensitive (e.g. the very fact a relationship exists between parties carries meaning) and thus care must be taken to not inadvertently distribute this information to the wrong or inappropriate parties.
* In the digital realm, a Proof of Relationship token may be needed. Consumers of such a token would require standard ways to validate that the relationship was still valid
* Generating Proof of Relationship information might require all parties in the relationship to interact in concert. For example, each member of the relationship has take an action in order to allow the release of Proof of Relationship information.
* Concepts such as User-Managed Access Control and Consent Receipt may be a portion of what is needed to implement a Proof of Relationship service

## Constrainable

Although a relationship exists, parties involved may want to impose constraints on the relationship. These constraints may describe acceptable behaviors of the actors in the relationship, approved use of data by the actors, and the terms under which the relationship is terminated. And in this way, the “Constrainable” design principle feels familiar to our everyday lives in the analogue world.

But in that familiar is a bit of a trap. One cannot assume that all of the actors in a relationship are capable of:

* Asserting their desired constraints
* Acknowledging constraints
* Enforcing constraints
* Being held accountable for failure to uphold a constraint

Consider a “smart” lightbulb. The owner may want to constrain what data the light bulb sends to its associated IoT platform, but the bulb does not provide such an affordance. In this case, the owner’s only recourse (other than not entering into a relationship e.g. not using the lightbulb) is to look for something else to enforce her desired constraints such as the IoT platform to which the light bulb sends messages. Acting in this capacity the IoT platform takes on the role of a relationship manager - an actor which is aware of the context of a relationship and can act upon the relationships and parties in the relationship. The concept of a relationship manager is discussed in more detail SOMEWHERE ELSE.

There is a further aspect of this design principle: context. Although an actor has put a constraint in place, that constraint may not always be enabled. Based on context a constraint may be enable or become relevant. In this way, the older design principle of “Contextual” because an aspect of this design principle. Contextual triggers turn on and off constraints based on the desires of the actors and potentially enforced by relationship managers or the actors themselves.

## Mutable

Relationships, like most things in the digital identity world, change over time. Different parties enter and exit a relationship. Attributes of those parties change over time. Attributes of the relationship itself can change as well. Thus designs for systems than handle relationships must account for mutability.

Designers should consider mutability at three levels:

* The relationship as a whole including all of the actors, constraints, and attributes
* The connections between parties and the associated attributes of those connections
* The actors and their associated attributes

Some aspects of a relationship may actually be immutable. For example, a connected device may be immutably stamped that it was built by Company Q. At the same time however, the connected between Alice and her connected light bulb may only last as long as Alice owns her apartment.

If change is inevitable, a fair question to ask is, “What manages changes to relationships, actors, constraints, attributes, etc?” Although individual actors may manage their self-asserted attributes, the IRM Working Group felt the need for a “higher level” manager, one who could enforce mutability across an entire relationship graph and delegate authority as necessary. As with “Constrainable,” the notion of a relationship manager appears.

## Revocable

Relationships end. This is true in the digital world as it is in the analog one. To “I am no longer in this relationship” may have a clear and distinct meaning to one party but not the other parties in the relationship. When discussing this design principle the IRM Working Group thought of it as equivalent to terminating a relationship and it quickly realized implementing relationship revocation was not as simple as just disconnecting the parties in the relationship. Questions arose about who can revoke a relationship and how is that revocation enforced, the preservation of historical information about the relationship, and the interplay of mutability and revocability.

How the revocation of a relationship works, what is required to revoke a relationship, and the process by which a party requests to revoke a relationship all differ based on context. Different industries and jurisdictions have their own interpretation of this design principle. For example, what it means to revoke Bob’s relationship with his smart light bulb is quite different from revoking Bob’s relationship with the country of his birth.

Guidance for designing systems that handle relationships includes:

* Consider legal and business requirements on the termination and revocation of a relationship.
* Coordinate data retention requirements with relationship revocation. For legal reasons, an organization may need to retain, long after the relationship end, proof of relationship as well as materials used to form the relationship and data produced from the relationship.
* Design a process for a party to request to revoke a relationship. (Design a process for reinstating the relationship too.)
* Clarify how revoking a relationship is different from changing attributes of the relationship or the parties in the relationship.
* Consider whether in the reader’s use case revocation is actually adding a broad constraint to the relationship.
* Given jurisdictional or business requirements, design the system such that revoking a relationship does not impede providing proof a revoked relationship existed in the past.

Given the influence of context on this design principle, the IRM Working Group did not delve into the specific mechanics of revocation. It is likely that the orchestration of business process, retention of records, etc are left to the notional “relationship manager” to sort out.

## Delegable

Relationships change. Relationships end. The actors in relationships can be replaced as well. In order to represent and handle situations in which the actors in a relationship change, either permanently or temporarily, relationship-based systems need to accommodate the design principle of delegation.

There are three areas of consideration for delegation and relationships: scope, permanence, and constraint. A party may choose to give another party all of its original capabilities and rights with regards to a relationship; in this case the scope of delegation is “full.” The original party may choose to impose no new constraints on the relationship meaning that the new party can do as they please in the relationship. The original party may be able to put a time limit on the delegation, stating that the new party has delegated participation in the relationship for 60 days, 100 hundred years, or it may be a permanent delegation.

For example, Alice may choose to delegate her participation in a relationship to Bob completely with no time limit and no constraints. Going forward Bob is linked to all of the other actors Alice was in the relationship and is subject to all of the existing constraints that Alice was subject to. An interesting question to ask is, is Bob entitlement to all of the historical data generated by Alice in the context of the relationship?

Another example, Bob delegates his participation in a relationship to Eve for 6 months and also creates a constraint that Eve is only allowed to observe data flowing in the relationship but not allowed to create new data. Meanwhile, Bob also delegates his participation in the same relationship to Alice for 30 days, in which she is granted full rights, except she cannot access historical relationship data and she cannot further delegate participation. This example begins to highlight the challenge of the delegable design principle in determining what is actually delegated: the party’s connection to the relationship or the party’s behavior in the relationship or both.

Notice the word “may” in the second paragraph. The IRM WG found it difficult to assert that actors in relationships would always have the ability to delegate their participation in a relationship. Furthermore, if a party can delegate their participation it is unclear that the party can always delegate the entire relationship for an indefinite amount of time without constraints. Depending on the context (including the legal context in which the relationship exists) actors can delegate differently. In some cases, the Reader can foresee that the other parties in a relationship may have a say in whether an actor can delegate their participation. Sorting out who can delegate, how much, and for how long is likely the job of a context-aware relationship manager.

## Scalable

Scalability is a must for identity relationship management. Originally, the IRM WG identified four axes of scalability: actors, attributes, relationships, and administration, and these four variables of scalability still need to be solved for in order to have relationship management. But there is another crucial consideration for this design principle - every party in a relationship may be legion. IRM is not only meant for single party to single party relationships, but also groups of actors in relationships with other groups of actors. As one member of the IRM WG stated, “this world is many to many on all sides of the equation.”

One way to think of the many to many nature of relationships is take a page from the Eames’ “[Powers of Ten](http://www.eamesoffice.com/the-work/powers-of-ten/).” Observing a relationship graph at an actor-level, one would see each individual actors connected to one another. Zooming out, one would see the organizations to which the actors are associated connected to one another. Zooming out again, one would see how the relationship graphs themselves link to other relationship graphs. Zooming in, one would see the attributes of the connections between actors and the of the actors themselves.

The practice of zooming into and out from a relationship can help the reader then recognize some of the challenges related to other design principles. At a certain “scale,” delegation becomes an organizational policy while at a smaller scale an individual actor may be unable to delegate their portion of a relationship involvement. At a certain scale, an actor may be allowed to change relationship attributes but at another those attributes are no longer mutable.

## What’s different?

Matter of scope - do these principles apply at the attribute of the relationship level or for the relationship as a whole.

Acknowledgeable is folded into Provable

## Acknowledgeable

Original text:

**Relationships can be acknowledged.** Participants can acknowledge that they have relationships to other actors. In this regard, the acknowledgeable characteristic of relationships feels very similar to single-party asserted relationships. A question worth asking is, “Must all parties in a relationship acknowledge they are in a relationship?” In a situation where only one party knows of the existence of the relationship, then there is an asymmetry of power. The party that knows about the relationship can exert some form of control over the other party. For example, credit bureaus acknowledge their relationship to me but do I acknowledge my relationship with them? Similarly, I acknowledge that I have a relationship with Twitter, but do I acknowledge my followers? Do my followers acknowledge a relationship with me?

It is interesting to note that rewriting the first sentence of the previous paragraph to read, “relationships must be acknowledged by other actors” leads to a discussion of Vendor Relationship Management scenarios and techniques. It also leads to questions of personal sovereignty and data ecosystems.

Transferable becomes Delegatable

Immutable becomes mutable

Contextual folds into Constrainable

## Contextual

Original Text:

**Relationships can be contextual.** More accurately stated, some relationships can be “triggered” by changes in context. Changes to conditions external to the relationship can have bearing on both how the actors in the relationship behave as well as what an external party can observe about the relationship.

Consider this example scenario: Before traveling abroad, I contract with a mobile network operator (MNO) to get a SIM card that will allow my phone to work at my destination. Until the SIM card via my phone connects with and pings a cell tower the relationship is inactive. The MNO doesn’t bill me for my usage because there’s been none. Once my phone with the SIM in it activates the relationship (by connecting to a cell tower at my destination) then the relationship between me and MNO springs into action and I begin to be billed for my usage.

Actionable dissolves into the world of the relationship manager

## Actionable

Original text:

**Relationships must be actionable.** We want relationships that are able to do something of value and, more specifically, relationships that can carry authorization data. However, relationships are not required to carry authorization data. The key is that they have the ability to do so.

In a traditional IAM scenario, we pass actionable information to the back-end for a classic request-response authorization model. But in an IRM (and IoT) world we must design for situations in which there is little to no connectivity to a back-end authority or that a back-end authority simply does not exist.

# IRM in the Wild - Sal to draft

# Paths to Follow

## Relationship Manager

## Relationship Notation Language

# Conclusions

# References

# Revision History