draft-wendt-modern-identity-registry-00

Modern Working Group
IETF97
Chris Wendt

Overview

- Documented a registry model that is compatible with TeRI concepts and objects but has some important differences
- Incorporates how the registry might be used in distributed environment as well as for provider internal objects
- Incorporates explicit service associations
- Introduces a model of public identity association(s) and routing identifiers as two related but separate objects.

Actors in this model

- **Provider** - An entity that provide a service to customers and manages there identity in the network.

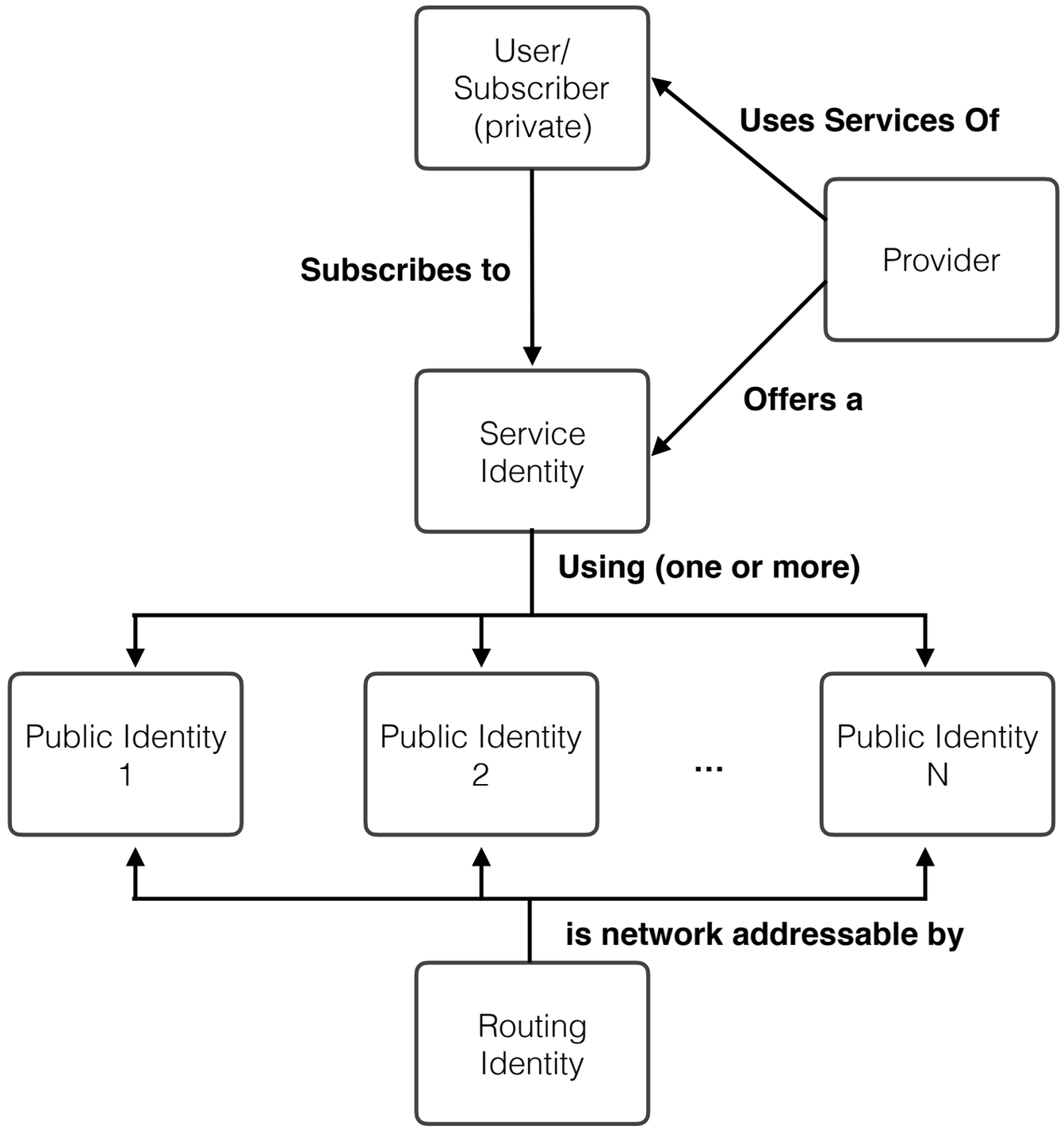
- **User/Subscriber** - The entity that is using the services of the provider.

- **Service Identity** - A globally unique identifier representing a application or service being made available to users/subscribers by multiple providers.

- **Public Identity** - A publicly known identity that the user associates with a service. This identity must be globally unique to a user/subscriber. It must also be provably associated to a given user/subscriber that claims the association.

- **Routing Identity** - A uniquely and globally routable identity used specifically in signaling calls between users.

Relationship diagram



Registry API functions

- The ability to query for available/unused identities for the purposes of either identifying conflicts before committing to the registry or identify unused identities that are part of a pool (e.g. telephone numbers)

- The ability to allocate identities for future use at individual levels or at block levels, such as NPA-NXX level telephone numbers or perhaps wildcard identities, e.g. *@example.com.

- The ability to update/transfer/port identities from one provider to another provider.

- The ability to digitally sign transactions to a provider for validation of legitimate transactions. Or forensic analysis of illegitimate transactions.

- Support of APIs to work with distributed registry (i.e. DRiP and GETs and PUTs to distribute changes to distributed registries).

Messaging and Control Flows

- Query Contexts
    - Typical queries for finding a globally routable identity should be in the context of a public identity and service identity for an allocated routing identity.
- Allocation/Assignment Global Uniqueness
    - When a provider customer has decided to allocate a given single identity or, for example, block level set of telephone numbers there is a PUT command that allocates the identity, given the number wasn't already allocated between the GET and the PUT. As a result of a successful allocation, the telephone number will be removed from the unallocated bucket.

Example Allocation/Assignment

- **publicID**: telephone number in e.164 format (e.g., +12155551212).
- **serviceID**: "voip" by default, other services potentially in future.
- **routingID**: SIP URI with telephone number + domain representing service provider of record (or perhaps spid) (e.g., sip: +12155551212@voip.example.com).
- **timestamp**: a timestamp retrieved from a common NTP server representing time of allocation, used for validating which service provider allocated first in race condition scenarios, and just for logging and historical reference in general.
- **x5u**: used for validation of signature
- **signature**: using a provider level [RFC5280] based private key/certificate, the provider MUST sign the information above to validate the change to the registry.

Update Entry/Port

- If a provider needs to update information related to an allocated entry, such as adding a publicID, modify routingID, etc. or if there is a port where a new service provider will overwrite the entry with new information, the API should be the same.
- There is a GET operation to read the current entry information, if the provider needs this information, (e.g., read/modify/write). There also is a PUT operation that will write the updated entry information. This will require a new timestamp and signature to validate the security of the operation and logging/historical purposes.

Removal/de-allocation

- If a provider wants to remove an entry for the case where a customer removes his service and no longer wants to own or associate a public identity, a DELETE operation will be provided that will delete the entry, and for the case of a telephone number, will put the telephone number back in the pool of unallocated numbers.