App-App Local Discovery Flow

Jeff Disher (Copyright 2013 Open Autonomy Inc.)
Date: 2013-08-22
Version: 1

Introduction:

The real power of OpenAutonomy is in the ability for multiple applications to work together. This document describes the "local" case of application discovery, which is the case where both requester and provider applications are managed by the same identity.

Capabilities:

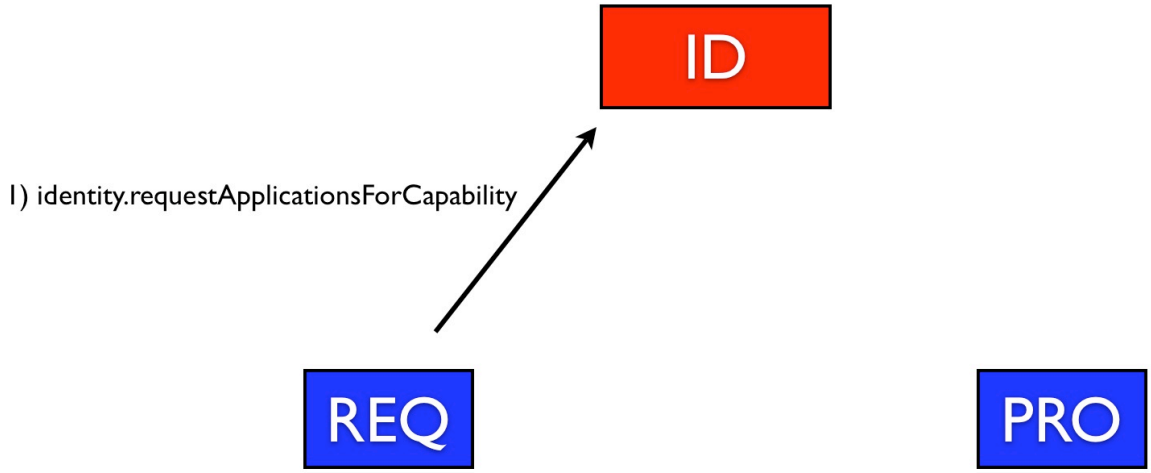
Applications find each other via advertised capabilities. If an application wants others to be able to find it to use its services, it advertises each service to the identity via its identity.advertiseApplicationCapability RPC method.

NOTE: Only an application with "MAX" trust will be allowed to offer capabilities.

As an outline, consider the applications REQ and PRO, as both the requester and provider of a given capability. Both applications are owned by ID.

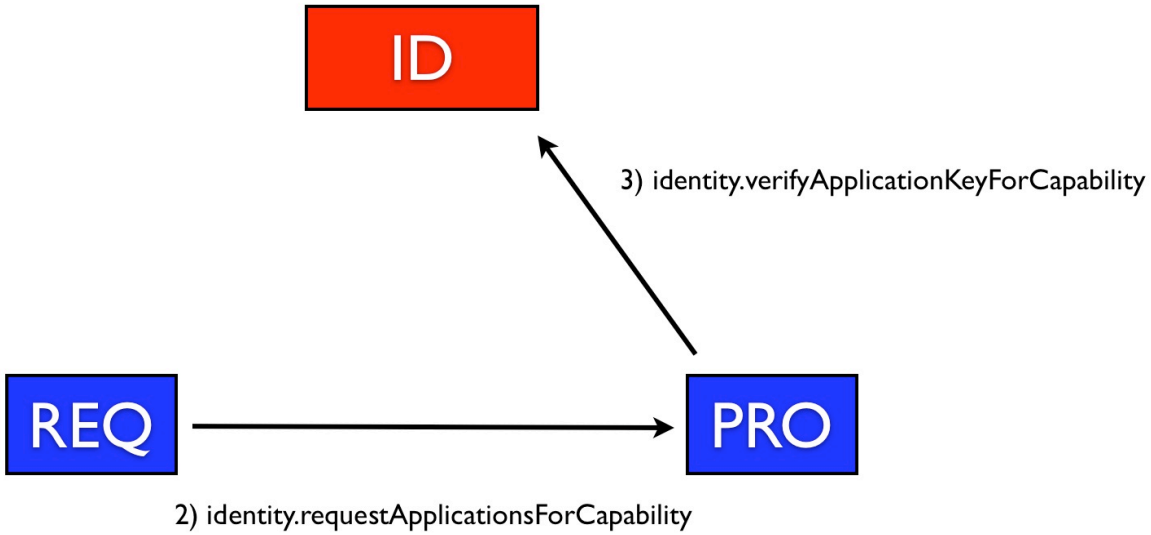
Steps:

1. REQ contacts ID via its identity.requestApplicationsForCapability RPC method. ID returns an array of tuples* describing the applications it manages which provide those capabilities.



2. REQ selects PRO from this array and contacts it via its application.requestRemoteAccess RPC method, passing in ID and the key given to it by ID.

3. PRO receives the request and, if the given identity is its owner, it contacts said identity via its identity.verifyApplicationKeyForCapability RPC method, passing the key it was given.



4. ID checks to see that this was the key it gave out for this PRO and returns a tuple** describing REQ (since it knows who it gave the key to).

5. PRO examines the tuple to determine which identity owns REQ and how strongly it trusts it. If it agrees to provide its services to REQ, it stores the key (advanced by one iteration) and the stride from ID for future validation of communication and returns true.

6. REQ now advances the key by one iteration and stores it, as well as the stride provided in the tuple from ID, for future communication.

7. REQ and PRO now have a key-stride combination for future calls between them (but these must only originate from REQ) and also know what their shared identity thinks thinks of the other in terms of trust level.

Tuple descriptions:

Both tuples are string->string associative mappings.

*(called “FullApplicationInfo”) contains keys:

- **trustLevel**: NONE, MIN, MAX

- **name**: Name given to the application by the user
- **url**: The URL where the application instance lives (similar to “identifier”, below, but MUST be a URL)
- **auth**: VERSION-KEY-STRIDE

**(called “RemoteApplicationInfo”) contains keys: trustLevel, name, stride, identifier, referringIdentityURL.

- **trustLevel**: NONE, MIN, MAX
- **name**: Name given to the application by the user
- **stride**: The STRIDE component of an auth key
- **identifier**: The unique identifier the identity uses to refer to the application (similar to “url”, above, but might not be a URL as this could be a “native” application)
- **referringIdentityURL**: In the case of local discovery, this is always the owning identity