Provisioning In Federated Environments

Phil Hunt, Oracle

January 2011

Introduction

- Initiated by Oracle and NSN (Nokia Siemens Networks)
- How to provision in SAML environments?
  - Customer at app store changing telco providers
    - Goal: Provision an IDP while de-provisioning another
  - Enterprise updating employee attributes in cloud services
  - Enterprise de-provisioning retired employees from cloud

Initial Approach

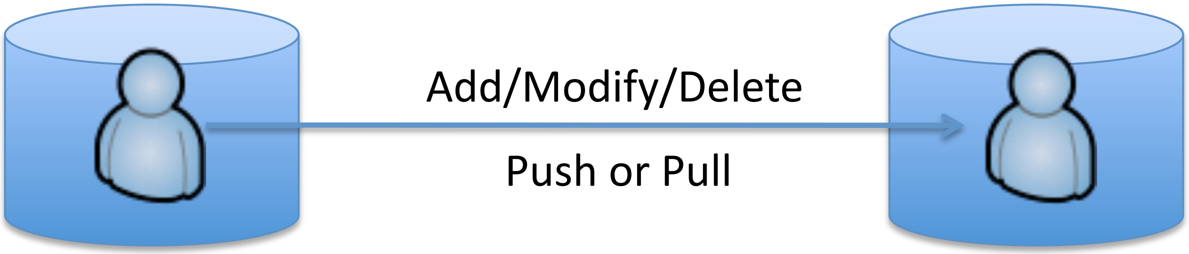
- Add CRUD operations to SAML
  – Good for SAML-centric relationships
  – Debate over SPML vs. SAML
    - SPML seen as yet another protocol
    - Desire to have single-protocol solution in some cases
  – Direct update with low-overhead
  – NSN SAML Attribute Management Proposal
  – Oracle Change Management Proposal
    - More capability, full attribute mgmt capability
  – BUT…

Challenges in Federation

- How to fit with some of the common web profiles of SAML (e.g. Web SSO Profile)
  - Provisioning convolved with sign-on (good and bad)
  - Get all the data needed at sign-on time
  - False Assumption:
    - no retained data == nothing to update!
- SAML has a provision on-the-fly approach for IDP-to-RP but not RP-to-IDP
  - Unless you assume role change in network
- Asserting party blind to actions by RP
- SAML has de-federation but no de-provisioning
- No detailed error reporting!

Federation IS Different

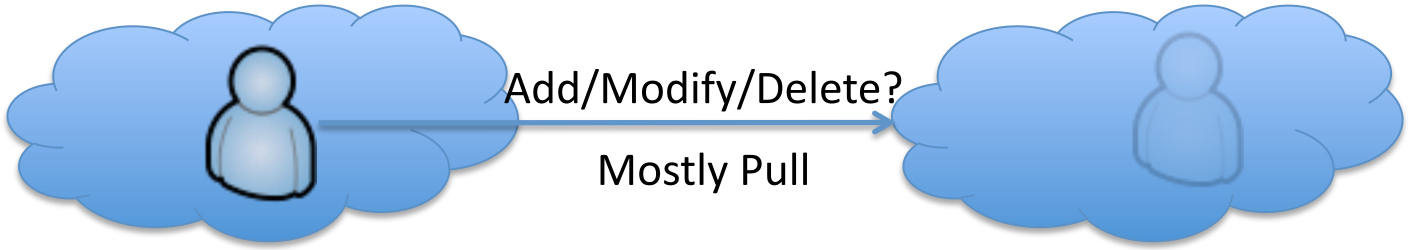
Internal Enterprise Provisioning



Add/Modify/Delete

Push or Pull

Federated Enterprise Provisioning



Add/Modify/Delete?

Mostly Pull

Matters of State

- Inside a corporate domain, entity state is known
  - Control of entities can be assumed
- Between federated domains, entity state can not be assumed and may be unknown
  - Entities can be influenced but not controlled

Observations About State

- Protocols/approaches that depend on knowing entity state may not be well suited to federated scenarios

- Protocols may be adapted, but require loose-coupling

Role of Context

- Context
  - Partners can agree in specific situations to take actions (e.g. transfer of account)
  - Meaning can be independently inferred
    - Why information is being exchanged
    - Ability to enhance/transform standard protocol operations (using a read to facilitate a write)
  - Agreements between specific service providers
    - Error recovery
    - Dynamic performance limitations (rate of change)

Notify and Act

- Change Notify establishes a notification step that occurs prior to an action step
- Notifies a target of upcoming changes
- Target may accept or reject changes
- Establish context for exchange
- Avoids error states that would otherwise emerge

Notification Step

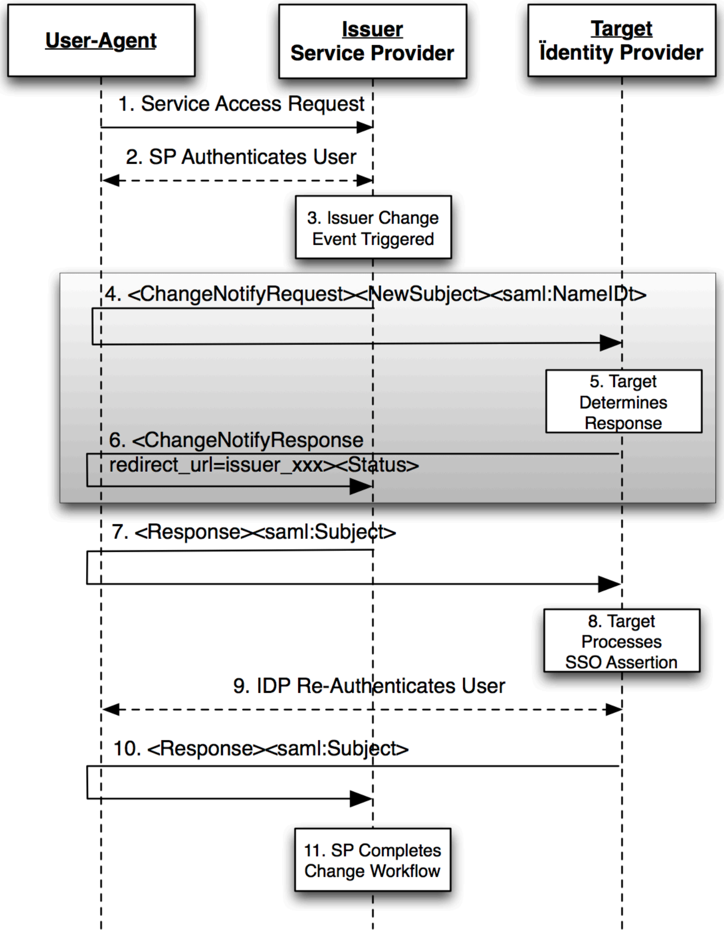
- Types
  - NewSubject – One or more identifiers which the notifier believes to be “new”
  - ModifySubject – One or more identifiers listing one or more attributes that are to be “changed”
  - RemoveSubject – One or more identifiers to be “removed”
- Notifications contain only identifiers
- Boxcar support – use of one or more identifiers allows message traffic to be reduced
- Can be used in online, front-channel profiles
- No claims / values transferred (except identifiers)
- Message SHOULD be signed

Action Step

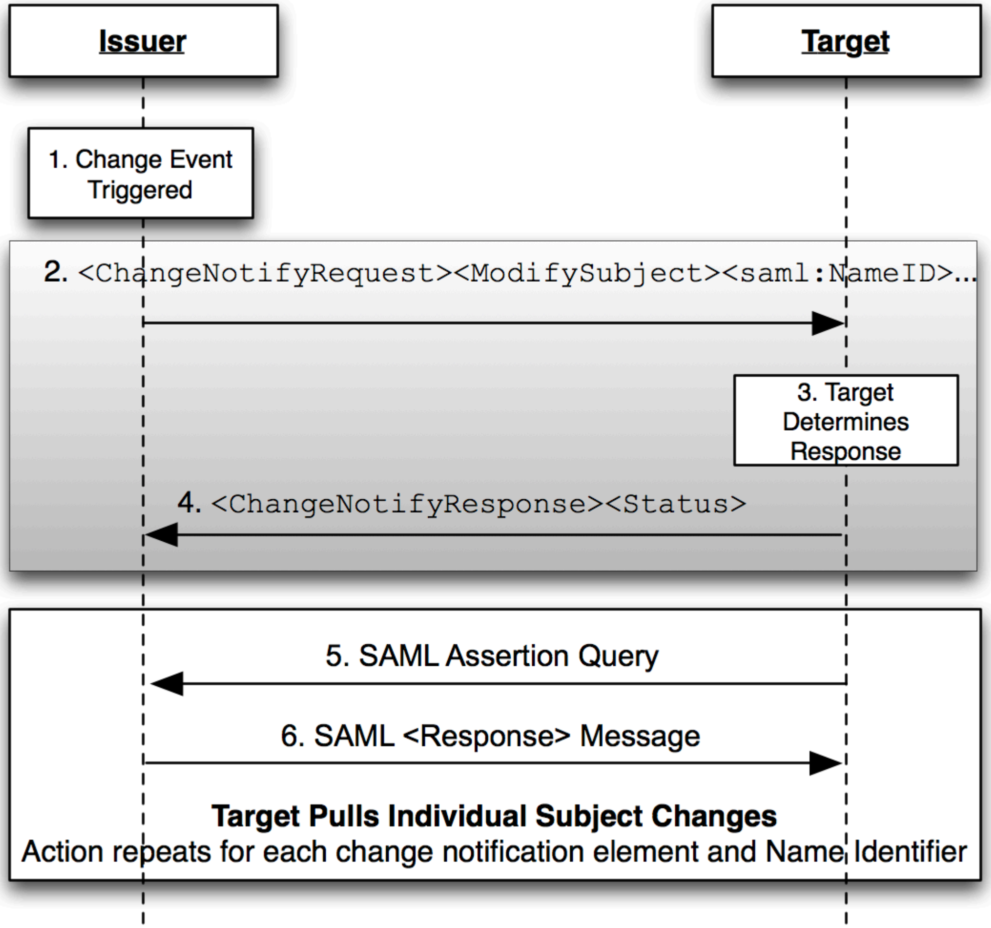
- Uses existing protocols to facilitate claims transfers
- May be PUSH or PULL
- Protocol could be almost anything:
  - SAML, OpenID, LDAP, SPML, PortableContacts, …
- E.g. NewSubject notification is followed by Web SSO profile to facilitate transfer of user information, ‘in-context’, and provide ‘warm introduction’

Examples

SP Initiates ‘Warm’ Registration


User-Agent
Issuer
Service Provider
Target
Identity Provider
1. Service Access Request
2. SP Authenticates User
3. Issuer Change Event Triggered
4. <ChangeNotifyRequest><NewSubject><saml:NameID>
5. Target Determines Response
6. <ChangeNotifyResponse redirect_url=issuer_xxx><Status>
7. <Response><saml:Subject>
8. Target Processes SSO Assertion
9. IDP Re-Authenticates User
10. <Response><saml:Subject>
11. SP Completes Change Workflow

Backchannel Update



Issuer

Target

1. Change Event Triggered

2. <ChangeNotifyRequest><ModifySubject><saml:NameID>…

3. Target Determines Response

4. <ChangeNotifyResponse><Status>

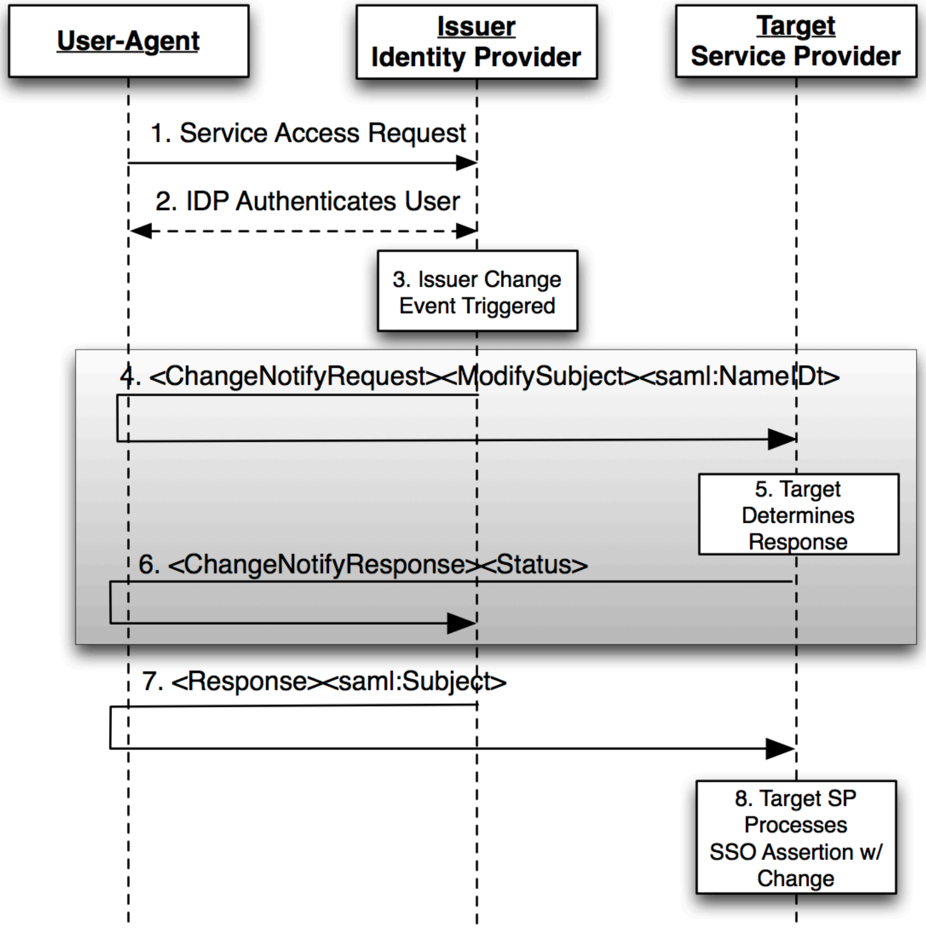
5. SAML Assertion Query

6. SAML <Response> Message

Target Pulls Individual Subject Changes

Action repeats for each change notification element and Name Identifier

IDP Initiated Change


User-Agent
Issuer
Identity Provider
Target
Service Provider
1. Service Access Request
2. IDP Authenticates User
3. Issuer Change
Event Triggered
4. <ChangeNotifyRequest><ModifySubject><saml:NameIDt>
5. Target
Determines
Response
6. <ChangeNotifyResponse><Status>
7. <Response><saml:Subject>
8. Target SP
Processes
SSO Assertion w/
Change

Example SAML Notify Request

```
<samln:ChangeNotifyRequest xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
    xmlns:samln="urn:oasis:names:tc:SAML:2.0:notify"
    ID="aaf23196-1773-2113-474a-fe114412ab72" Version="2.0"
    IssueInstant="2006-07-17T20:31:40Z"
    protocol="urn:oasis:names:tc:SAML:2.0:notify:protocol:saml:FrontChannel" >
    <NewSubject>
      <saml:NameID
           Format="urn:oasis:names:tc:SAML:1.1:nameidformat:X509SubjectName">
           C=US, O=NCSA-TEST, OU=User, CN=john.doe@corp.com
      </saml:NameID>
      <saml:Attribute
           xmlns:x500="urn:oasis:names:tc:SAML:2.0:profiles:attribute:X500"
           x500:Encoding="LDAP" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
           Name="urn:oid:2.5.4.42" FriendlyName="givenName">
      </saml:Attribute>
      <saml:Attribute
           xmlns:x500="urn:oasis:names:tc:SAML:2.0:profiles:attribute:X500"
           x500:Encoding="LDAP" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
           Name="urn:oid:1.3.6.1.4.1.1466.115.121.1.26" FriendlyName="mail">
      </saml:Attribute>
    </NewSubject>
</samln:ChangeNotifyRequest>
```

Response

```
<samln:ChangeNotifyResponse xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
    xmlns:samln="urn:oasis:names:tc:SAML:2.0:notify"
    xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
    ID="aaf23196-1773-2113-474a-fe114412ab72" Version="2.0"
    IssueInstant="2006-07-17T20:31:40Z">

    <samlp:Status>
     <samlp:StatusCode Value="urn:oasis:names:tc:SAML:2.0:status:Success" />
    </samlp:Status>
</samln:ChangeNotifyResponse>
```

Status of Proposal

- Currently a SAML Protocol Proposal
- Written by NSN and Oracle as part of the OASIS Security Services Technical Committee
- Working Draft 04
  - – http://www.oasis-open.org/committees/document.php?document_id=40036
- Voted to Committee Draft status

History

- Working Drafts 01, 02
  - Exploration of push model: Add, Modify, Remove
  - Primary problems became
    - Error handling
    - Need to quantify entity state
- Working Draft 03
  - Evolution to 2-step
  - Push notification followed by negotiated multi-protocol action step
  - Boxcarring permitted
  - Issues
    - How to handle name identifiers for multiple protocol choices
    - Too much negotiation
- Working Draft 04
  - 2-step
  - Push notification followed by pre-negotiated protocol step
  - Simplification
    - No in protocol negotiation of “action” step – but can be achieved
    - Per protocol end-points
    - Identifier handling
    - Single multi-purpose front-channel and back-channel profile

Future

- Should Change Notify exclusively be a SAML Protocol?
- Is there interest in exploring a lightweight variant?
- Profiling ChangeNotify and SPML