Data Manipulation

Jonathan Rosenberg
dynamicsoft

XCAP Issue #1

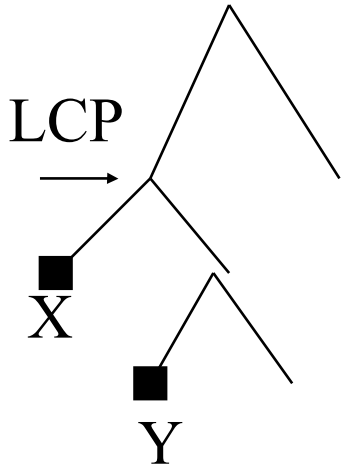
- Should we just instead use actual filesystem hierarchy for buddy lists?
  - I.e., each buddy is a separate file
  - Makes it easy to modify individual buddies with HTTP
- Issues with that:
  - Want buddy list to exist as a single document, to facilitate
    - Client side storage
    - Transfer
    - Transformation (I.e., generate an HTML page with my buddies)
  - May still need XCAP to do XML element addressing anyway

XCAP Issue #2: Batching

- Perceived requirement: The ability to make multiple changes that are atomic
  - Multiple changes may be needed to achieve a doc that is schema compliant
  - An intermediate state may represent undesirable behavior
    - A user on neither an allow or deny list

Batching Options

- Soln I: Least Common Parent
  - If changes need to be made at nodes X and Y, read the least common parent (LCP), make the change, and write
  - +: easily done in XCAP
  - -: wireless efficiency really bad



Batching Options

- Soln II: Body Commands
  - Make XCAP SOAPISH by placing the operations as XML in the body
  - +: easily does batching, efficient
  - -: not simple anymore
  - -: may be replicating other work (I.e., webdav)
  - -: violates rfc3205

- Soln III: Webdav
  - Use its versioning capabilities along with LOCK/COPY/modify/ MOVE/UNLOCK
  - -: May require webdav changes to deal with partial documents
  - +: reusable for webdav
  - -: may take some time

Issue 3: Server awareness

- Currently, the spec says a server needs to understand the application usage against which requests are made
  - That is, server needs an upgrade for a new app
- May be possible to lift this for application usages which
  - Have no computed data
  - Have no additional data constraints
  - Follow the baseline authorization policy
- Do we want this?

Issue 3.1: XML Extensibility

- Application usage defines the schema, which the server needs to know
- What if schema defines extensibility, and a user adds data outside of the defined schema, using a namespace/schema not understood by server?
- Proposal: direct extension of previous issue – server needs to understand all of the namespaces

Issue 4: Server Authorization

- In ACAP, authorization was built into the protocol
- In XCAP, I am proposing that there is a trivial default authorization policy
- If you want a more complex one, you need an application usage to represent the authorization policy
- This really simplifies the protocol a lot
- Is this constraint OK?

Issue 5: Insert vs. Modify

- Current document uses POST for insert, PUT for modify
- Doesn’t seem right
- We need both – how to do it?

Some important observations

- Anything other than an obvious usage of HTTP will require much broader input
  - Design team as suggested by Ted
  - Add +1 year of time
- What's important to us is the SCHEMA, less so how it gets transferred and munged

My Proposal

- Descope XCAP so that it is nothing more than an HTTP Usage (more on what that means later)
- Focus on the schema design
- Work XCAP v2.0 with WebDav to add new features

Implications of HTTP Usage

- No batching
- No locking/unlocking
- No POST – PUT only
  - PUT to a node that exists means modify
  - PUT to a node that doesn't means insert
    - Where its inserted is up to the server within schema constraints
- Partial document modification using Xpath URI
- No server computed data or data constraints

How do we get around these limitations?

- Carefully design the schema so that you can GET/change/PUT useful subsets in one operation
  - For auth policy, its not white lists and black lists, it's a list of users, and for each, a list of permissions
- Carefully define schema so that inserts can be done in places where they are needed

Data Manipulation Requirements Changes

- Added a requirement for display name property on resource lists
- Added a requirement on list data extensibility
- Limited the scope of authorization policy to presence
- Acceptance requirements based on domains and wildcards
- Notification requirements from MUST to SHOULD
- Can specify tuples a watcher should get based on attribute

• Different watchers get different information by presentity publishing different info

• Consistency requirement generalized – doesnt require batching

Data Manipulation Requirements Proposed Approach

- Submit in parallel with xcap drafts
- Avoids waterfall requirements process
  - We can adapt requirements based on protocol mechanism

Authorization Usage Structure

- Authorization is a set of <statement>
- Each <statement> has an <applies-to> that specify who the policy applies to, and then a series of permissions
- <applies-to> can specify a URI, a domain, or a list
- Each permission grants the ability of a watcher to get or do something
  - Permissions are POSITIVE – you are allowed to do something. Not NEGATIVE. This makes for easier composition and allows schema to be edited more easily

Authorization Usage Structure

- Permissions in several classes
  - Acceptance: <accept> and <accept-if>. <accept> gives permission to subscribe. <accept-if> gives permission if the embedded boolean expression is true.
  - Boolean expression gives conditions on request – subscription lifetime, authentication mechanism, can-encrypt, filters

Authorization Usage Structure

- Rule Permissions
  - Specifies event transitions that watcher is permitted to see
  - <any-event>: All transitions
  - <enter-state>, <exit-state>: entering or leaving specific state
  - <change-in>: certain attribute changes
  - <equals>: send notifications if a certain attribute has a certain value
    - I.e., send notifications to Joe if my placetype is home.

Identifying Presence Data

- Some of the permissions are based on presence or value of an element
  - Placetype is home
- Requires the ability to identify a specific XML element
- Two ways
  - By element name: Refers to any element with that name, in the document or as input to composition
  - By Xpath: Refers to a specific element in post-composed document

Content Permissions

- What is the watcher allowed to see when they get a notification?
  - <all-content>: everything
  - <show-tuple>: show a tuple by name
  - <show-namespace>: show elements in a namespace

Transformational Attributes

- Modifications to the document that get made for watchers
- <set-document>: send them this document
- <set-element>: set this element
- <change-element-from>: if an element has a value, change it to this value

Open Issues

- Union vs. Most Specific
  - If multiple statements match a watcher, do you union the permissions or take the most specific
  - Proposal: union – consistent with intra-statement overlaps

- Eliminate applies-to?

– Can do it with <accept-if>, and adding conditionals to other permissions

– Proposal: No. Applies-to makes schema more amenable to transaction-less editing

Open Issues

- Identifying elements
  - Is the approach in the document right?
  - Need to think about it a bit more
- Are people happy with the scope?