Messaging at University of Washington

The title is a bit general. This is how we do some AWS SNS/SQS messaging, in particular the provisioning of group systems downstream from our registry: AD and Google groups.

See my 2¢ on message content.

UW's group resources

We have essentually two types of group resources: group general information, and group member. The general information resource contains everything but membership. For this discussion the member resource is a group-id/member-id tuple.

Our web service also rolls all members into a single membership resource, but that isn't relevant to this discussion.

Generation of events

Our group web service, via grouper hooks, watches update activity and records two types of events in a **uw_activity** table in grouper's database:

1. Group info has changed.
    a. event time
    b. group's id
2. Member added or deleted.
    a. event time
    b. group's id
    c. member id
    d. add/delete

Of these member add or delete is by far the most common event.

Generation of messages

A daemon process continually reads the **uw_activity** table. It accumulates events for each group until there is no activity for three seconds for that group. At that point it:

1. Updates our openldap directory. This ldap is used as a cache for most membership queries and needs to be quickly up to date.
2. If the group has changes (type 1 events):
    a. GET a representation of the group
    b. Send the representation to the SNS topic.
3. If the membership has changes (type 2 events):
    a. Send a member representation to the SNS topic.

In either case message header information distinguishes PUT from DELETE.

Specific message structure

All messages are base64 encoded and delivered to AWS in the usual way. A message contains a standard header and specific body content: (example message)

Header

```
{
 "header": {
  "version": "UWIT-1",
  "contentType": "xml",
  "messageContext": "some-base64",
  "messageType": "gws",
  "messageId": "f69409e4-9c5b-3c7c-a401-89297e9a1320",
  "sender": "gws",
  "timestamp": "2013-12-09T17:10:21.049Z",
  "signature": "some signature",
  "signingCertUrl": "https://groups.uw.edu/pubkeys/sign1.crt",
  "keyId": "iamcrypt1",
  "iv": "s+irb/dkXIvnh94cBB0ZLA=="
 },
  "body": "base64 of the message"
}
```

1. The version identifies the signing and encryption algorithms.
2. The signature is a standard hash of a concatenation of parts of the message---similar to AWS signatures.
3. Presence of keyId and iv means the body is encrypted.

messageContext

The context is application specific. For groups, for example:

```
{
  "action":"update-members",
  "group":"course_2014win-french102i",
  "actors":{
    "id":"urizen3.cac.washington.edu",
    "as":"root"
  },
  "time":1386021063895,
  "targets":[]
 }
```

Body

The body is a representation of the resource. For update-members: (roughly)

```
<gws class="gws">
 <header>...</header>
 <group class="group">
  <regid class="regid">group's regid</regid>
  <name class="name">group name (cn)</name>
  <add-members class="add-members">
   <add-member class="add-member" type="{type}">{member_id}</add-member>
   <!-- more member if multiple members are added -->
  </add-members>
  <delete-members class="delete-members">
   <delete-member class="delete-member" type="{type}">{member_id}</delete-member>
   <!-- more member if multiple members are deleted -->
  </delete-members>
 </group>
</gws>
```

The 'odd' style is historical, a combination of xml and xhtml.