Minutes of the 8-29-2011 concall

OSIdM4HE-prov concall, 8/29/2011, 16:00 - 17:00 ET

- **Attending:** Rob Carter, Keith Hazelton, Lucas Rockwell
- **Minutes of the 8/19 call:** Attendees will review and forward Rob any changes. Rob noted that the minutes for this call may be sketchy based on his limited notes. Meantime, notes are posted to the team workspace on spaces.at.internet2.edu.
- **Agenda bash:** Nothing of significance
- **Action Items from 8/19 call:**
    - Rob noted that he'd added some text in a third column to the architecture diagrams page walking through a couple possible scenarios for processing provisioning events.
    - Lucas noted that he'd sent out responses to his two action items just before the rescheduled call on 8/26. Rob apologized for missing the email on Friday. Both these items and the preceding item are on the agenda under New Business.
- New Business
    - Attendees spent a few minutes reviewing the new material, since Rob only posted it minutes before the call…
    - Keith noted that while the detail in the existing diagram is something we'll need to work with before actually committing to an implementation plan, it may be more low-level than will be appropriate or effective for certain audiences. He pointed out that Lucas's diagram takes a much higher-level view of things, which may be more appropriate for some audiences, and suggested that it might be interesting to try mapping the problem space onto enterprise patterns from the enterprise integration patterns book.
    - Lucas agreed that the primary dichotomy between the two diagrams is around the level of implementation detail
    - Keith agreed to take an action item to construct what he expects may be a "middle" option architecture diagram using components from the enterprise integration patterns book
    - Rob noted the action item and agreed that such an effort would be well worth the trouble.
    - Keith noted that it may be interesting to consider how to move from one level of abstraction to another in presenting our material, and to consider how the layers of abstraction actually map onto real elements in a solution model – something that hasn't been very thoroughly explored in our circles, but that may be worth thinking further about.
    - Lucas raised the question, based on our discussion of audience and levels of abstraction, what level of abstraction might be appropriate for our mid-September deliverable.
    - Keith suggested that from the perspective of his strategy group, they would be looking for stuff to feed into a funding cycle – estimates of the amount and type of work required to solve the problem, scale of the problem, scope of the work, and how it fits into the "bigger picture"…
    - Rob suggested that it might be useful in addition to looking at higher-level abstractions of the provisioning problem itself to look at even higher-level abstractions of the entire IDM stack, focusing on the interfaces across the boundaries between the registry layer and the provisioning layer and between the provisioning layer and consumer systems/services. He noted that such a diagram might be a roughly the level of abstraction of the existing SCIM scenarios.
    - Keith agreed that this might be helpful, and noted that that's similar to what the canonical I2/Educause IAM diagram does.
    - Rob asked if Lucas would like to describe his diagram at this point, to move into the next part of the agenda.
    - Lucas explained that in thinking about diagramming our space at a somewhat higher level, he was struck by our earlier discussion of how the registry group might conclude their efforts and what needs they might express for services from our effort, but also what the registry group might expose that we'd want to take advantage of. He explained that he started with a goal, among other things, of making our provisioning position as independent of the details of the registry underlying it as possible – supposing that so long as the registry provides some mechanism for detecting and exposing changes, our solution should be able to interoperate with it.
    - Lucas outlined the diagram in broad terms – starting from the top: When the registry detects a change and records it in some fashion, the provisioning layer recognizes the change (either because it's explicitly delivered in some form of message or because it's watching for entries to appear in some form of changelog or changelog analog). Depending on the circumstances, the provisioning layer may be responsible for collecting both information about the existence of changes in the registry and specific change information from the registry. Once collected, the provisioning flow transfers to a messaging layer (depicted in the diagram as a JMS queue, but any similar sort of implementation could work as well) that incorporates persistence and has the ability to poll the registry as necessary to re-acquire change details (or marshall additional data to enact a change). Data flows from the messaging layer through one or more outbound interfaces to consuming systems.
    - Lucas noted that his preference would be for the smallest number of well-standardized outbound interfaces as feasible, and suggested that no matter what outbound interface protocol(s) might be chosen, we should consider trying to reach out to some of the most common higher ed oriented consumers (Sakai, Blackboard, Moodle, Google, PeopleSoft, etc.) and work toward convincing them to implement standard consumer interfaces (eg., SCIM consumers, etc.).
    - Keith noted that there's a core notion in all of our discussions so far of some form of "event" happening in the person registry and another of "downstream" systems being manipulated to reflect the "event" in their individual contexts. He pointed out that there seems to be no (or at least, no obvious) mention of such an "event" model in the SCIM documentation – there are well-defined transactions – add a user, remove a user, update a user – but nowhere in the SCIM model to manipulate an "event" per se. He wondered if perhaps he'd missed something in the documentation, since the event presentation seems pretty seminal.
    - Lucas noted that while the SCIM protocol itself doesn't really identify an "event" as something it can manipulate or use, there may be some mention of events in some of the documentation (perhaps as causes for transactions being processed).
    - Keith agreed and pointed out that the SCIM protocol seems to implement the standard GET/PUT/UPDATE semantics, but not any mechanism for posting an event or message of any sort. The payload in SCIM transactions is always, apparently, an entity object. He noted that it may be worthwhile for us consider different "weights" of interface between the registry and provisioning layers, and to try and determine what the absolute "skinniest" atomic event we could reasonably employ between the layers of the IDM stack might be. It might be a simple reference to an object ("Hey – something changed in object X – go do something if you want"), or as complete as a full entry rehash ("Hey – here's an object I just updated – make sure you have it and it's aligned with this one"), or a more complex operational event ("I just updated the value of attribute A1 in object O1 from value V1 to value V2").
    - Keith went on to note that in these cases, the usual response within higher ed to the question of what level of complexity to employ is that any level might be optimal depending on the specific use case. That being the case, he reasoned, it may be just as well that we implement the most bare core we can on the premise that it can be used to implement a richer set of semantics in cases where that becomes desirable or necessary. Perhaps, he suggested, there are ways we might think outside our usual box in this space, as well.
    - Rob noted that in his experience with both Novell and Oracle's IDM solutions that had been clear. Oracle's approach tends toward the lighter end of the protocol spectrum, passing "create/delete/update" events for specific attributes and objects out to its connectors, but only sometimes passing additional information about the value or values involved. Novell's approach tends to be just the opposite, with almost all changes being delivered fully clothed in attribute:value pairs.
    - Keith agreed, and noted that they were both originally (in one way or another) built around SPML or something similar, and that SPML has to some extent conditioned the development of SCIM with it's "here's everything – you figure it out" approach.

        - At this point, Keith's phone connection dropped him off the call.
    - Lucas reported that in reviewing some additional SCIM documentation, he'd found references to triggering events based around the traditional "add/modify/delete" model, but also noted that it seemed to always be coupled with additional context information from outside the event itself.
    - Rob noted that the SCIM scenario diagrams may have events implicit in the arrows between the boxes, somehow…
    - Lucas pointed out that in his diagram, arrows pointing into a box could imply reading or writing data directly from or to a system, or could imply something posting some sort of event from one box to another. He conjectured that it's really a question of where certain logic is implemented, more so than a question of whether certain logic is implemented at all.
      ***At this point Keith's phone connection was restored.
    - Rob suggested that a fruitful discussion might be built around the question of where the "logic" should reside (or what logic should reside where) in the larger overall IDM stack diagram. He suggested there may be certain logic (eg., attribute promotion logic for deciding which of a number of equivalent but non-equal values for an attribute provided by different source systems should take precedence for a given individual) that's best implemented in the registry, while other logic (eg., consumer-specific attribute mappings) might best be applied in some sub-layer of the provisioning layer of the stack. Defining what parts of the overall required set of logic get implemented in which "block" of the block diagram will be critical.
    - Keith indicated it may become interesting if we come up with a list of things we believe provisioning should be responsible for and a list of things we believe the registry should be responsible for, then compare with the same estimates by the registry team. We may find that there are gaps not covered by either team's thinking, or that there's overlap between the two. He noted that traditionally, we've considered the registry to be responsible for the entire space, but now that we've conceived of a multi-layer model with provisioning as a separate layer, there may be traditional registry functions that the registries folks may wish to have our group reflect on instead.
    - At this point, the clock ran out for our conference call, and Rob reiterated the action items for the next call, which is scheduled for Friday afternoon, 9/2, from 16:00 to 17:00 ET. Action items for the next call:

1. **Rob and Lucas will work on getting Lucas's diagram into the wiki along with some explanatory text**
2. **Keith will put together a diagram and some text decomposing a provisioning solution into blocks based on the enterprise integrations book's patterns**
3. *Rob will take a stab at devising a "50,000-foot" level diagram of the entire space, highlighting the interfaces between the registry and provisioning layers and between the provisioning layer and various consumer systems.

- Adjourned until 9/2.