Minutes of the 11-09-2011 concall

11-09-2011, 3pm - 4pm ET

Attending: Rob Carter, Lucas Rockwell, Keith Hazelton

Minutes of the last meeting: No dissent or corrections

New Business:

- Rob started the discussion by focusing the attendees on the list of questions framed during the last call that he'd forwarded to the mailing list for possible advancing on to the OSIDM4HE-Registries group, asking if the group might be prepared to send them off for consideration by the Registries folks.
- Keith noted that some of the Registries questions we'd identified might be amended or informed by discussion of the longer list of framing questions Rob had forwarded to the list earlier in the week as a starter for discussion of how to construct an actual proposal back to the main group for a provisioning "project".
- Rob agreed and suggested that the agenda be inverted to place discussion of the framing questions first, followed (time permitting) by the Registries questions.

- Starting from the top, the group then began reading through the questions and the answers provided to date...

- **Question #1: Should "provisioning" be inclusive of only one data source (registry) or multiple registries, and should it include only "identity" data or all data that an identity consumer may need from authoritative sources?**
- Lucas suggested that groups and permissions (and other things potentially considered for provisioning in this context) are derivable from existing data in the registry, and so can be considered part of the larger scope of "identity information" but that other data (not directly derived from or dependent upon classical identity information) should probably be excluded, since it would tend to both blur the concept of provisioning and lead to really difficult problems in handling data synchronization.
- Rob asked if there was general consensus on that point?
- Keith suggested that arbitrary non-person-related data is out of scope, but that the approach taken to provisioning should be flexible enough to handle non-person-specific or non-person-oriented data if needed -- handling that data simply isn't important for the provisioning tool. He gave persons, groups, and permissions as examples of person-oriented data that should be considered in-scope for provisioning, and things like router configurations as non-person-oriented data that should clearly be out-of-scope. He suggested that role-based accounts (eg., identities associated with a business role rather than an individual person) might go in either direction, but probably fit well into the person-oriented data model and should be considered in-scope.
- **Question #2: Should "provisioning" encompass just the creation of domain-specific user objects based on registry information, creation and maintenance of those objects, and their attributes, or management of the full object lifecycle (creation, management, and deletion or deactivation)?**
- Lucas noted that Keith's third option and Rob's proposed answer to this quetion were roughly the same. He explained that he agrees that if a provisioning system creates a domain-specific object, it should manage the full lifecycle of the object, from creation through attribute management and deletion or deactivation. He suggested that "fire and forget" models are a bad idea, and also noted that there will be cases in which objects are created outside the provisioning system, and may not be managed (nor even manageable) by the provisioning system.
- Keith proposed that the group might be over-focused on "provisioning" and suggested that there's a more general problem of which classical provisioning is just one facet. The definition of "provisioning" is paramount in this discussion -- and the whole lifecycle management of digital entities seems like somethin go fa superset of provisioning. Creation and attribute maintenance are fairly general rule-based data integrations, really -- state changes in some upstream system are exposed through the registry and operated upon using rules of arbitrary complexity that transform the state changes into other contexts. If a new hiring decision is made, for example, a cascade of events should happen. It may not be possible to reduce that operation to any higher-level rule description than "these are the things that should cascade down the institutional infrastructure in the event of a new hire", in which case, given that more than one ERP may be involved in any given individual case of such a high-level event, most of these events may be in-scope for a provisioning facility. He indicated that his argument would be that it's a good idea to think about the problem in this way -- in terms of data integration and cascades of consequences from high-level events, noting that if we start from that proposition and then limit the scope of the project, we'll at least be aware of what we're declaring out-of-scope (rather than possibly ignoring something important in an effort to positively define what's "in-scope".
- Lucas (to make sure everyone was on the same page) restated Keith's suggestions. So, he suggested, if you're talking about the registry and provisioning, say a new hire happens -- that may trigger a bunhc of events that may or may not affect the registry -- some of them may not involve any change in classically-defined identity data. The registry may get data about a new hire, and that may trigger the creation of a new person identity. Rules may then say the person needs a new account somewhere, which the provisioning engine would then create not because of the hire (directly) but because of the registry change.
- Keith agreed with Lucas' recapitulation, and noted that in view of those points, the provisioning engine's job can be defined fairly succinctly as making sure that all changes to person-relevant data percolate to the right places within the infrastructure. The complexity, it seems then, is in the writing of rules to drive the engine, and in selecting and implementing the right data transformations between source systems (be they ERPs or the registry itself) and target systems. It seems, he mused, that we keep coming back to the question of where in the global picture business logic should be encoded, and asked if it would be crazy to consider refactoring our thinking to pull business transformation logic out as a component all its own (separate from provisioning)? To consider business logic as a separate gap in need of filling by the OSIDM4HE. Would, he wondered, doing so reduce the debate around the issue, and would it perhaps be something that the registries team and the provisioning team could do jointly?
- Lucase noted that it may come down to thinking of the space a bit differently -- thinking of "provisioning" as the business logic and rules, separate from the "plumbing" that moves data between the registry and targets systems.
- Rob agreed that splitting out rules creation/modification/implementation as a separate service would make sense, but expressed concern that the group not end up reproducing the N^2 problem by adding additional interfaces into the mix.
- Lucas explained that in ITIM (IBM's IDM product) the "registry" is essentially a dumb database that knows only that a given connector exists and wants information about changes to certain attributes. The registry then hands changes (unfiltered) to connectors, which in turn know what to do (or ignore) with the change information.
- Rob explained that OIM has a similar model
- Lucas offered to discuss it with some of his colleagues who are closer to their existing provisioning interface and report back to the group at a later time.
- Keith noted that the concern about having multiple interfaces is a valid one, but hoped that by taking a compatibility-based approach to building business logic into registries and provisioning engines, it ought to at least be possible at some future time to talk about extracting business logic into some form of rules engine.

- Rob agreed, with the caveat that (in his opinion) there may be some business rules that are affine to the registry and other rules that are affine to target systems -- essentially, rules about when and how things should be provisioned (which are affine to the registry) and rules about how data should be presented to target systems (formatting, data transformation, etc.), the latter being more affine to each target system.
- Keith remarked that that would suggest a sort of enterprise view of entities, and a need for a sort of global shared enterprise view of data that individual target connectors can vary from if the want -- a standardized representation of person data with explicit, target-specific customizations. He noted that it starts to be reminiscent of SAML in a sense - crazy stuff goes on in local system, but in the end ,they all end up abstracted into one of a small set of well-defined SAML attributes. SAML implementations push in that direction, as do, he suggested, SCIM implementations. He added that taking that approach -- presenting a single shared enterprise view of person data through the registry and providing for customization with target-specific data transforms that map the standard enterprise view onto target-specific attributes and formats -- clarifies the question of who is responsible for what, as well. If the registry is dealing with a very customized view of the data presented to it by a specific source system, it would be the registry's job to normalize the data before making it available (in the common format) to the provisioning engine. In essence, each player in the game becomes responsible for consuming data in whatever format it's presented to them in, but providing it to other components in an agreed-upon, common format.
- Lucas agreed strongly with the idea of "common representations between systems, unique representations within systems".
- Rob noted that this part of the discussion implicitly seems to be tending toward a sort of event-driven model for provisioning.
- Keith agreed that that seems to be the endpoint his thinking keeps arriving at, as well
- Lucas also agreed, noting that in any other approach, there's a good chance that the whole process becomes "a huge registry hairball".
- Keith noted that the model begins to become "scarily" like a generalized data integration model.
- Lucas agreed, noting that such an event-driven model works both for tightly-coupled cases (such a Kerberos KDCs, which are probably most often run by the same groups that manage other identity services) and loosely-coupled cases (such as mail systems, which are typically not run out of the same organziational units as identity services). With a common data view in the core and customized data views at the periphery, it becomes possible for loosely-coupled systems to make their own decisions about whether to implement a "pull" approach (by consuming the common view of data provided in the core directly) and manage their own lifecycle arrangements, or outsource the lifecycling and arrange to have a provisioning engine "push" data to them, which they can then transform into their domain-specific formats as necessary without losing the benefits of outsourcing the lifecycle management components.
- Rob noted that this would also point out a need in the provisioning space for interfaces to allow folks outside the traditional IDM facility (out in the units responsible for those loosely-coupled systems and services) to access and update the rules by which their loosely-coupled systems receive provisioning events.
- Keith agreed, saying that at Wisconsin, they have a system that no one is willing to touch that's full of rules that no one knows how to interpret or use. It's essentially a problem of fossilization of business rules over time that somehow needs to be avoided, probably by surfacing as much of the internal business logic as possible, possibly even by extracting it into a separate service, where it can be managed by itself. Human nature in this case works against us, he noted, in that people seldom write documentation before code, and folks seldom define rules adequately before implementing them. To the extent, he noted, that systems can make it easier to document before implementing, we'll at least have a chance of combatting the failings of human nature.
- Rob mused that a near-natural language interface suitable for use by business owners would be optimal, but perhaps unattainable.
- Lucas suggested investigating RSpec for Ruby, which he suggested allows users to write english-like statements from which declarative rules are then derived -- it can, for example, convert statements like "when someone does X to Y, Z should happen" into actual code-manipulable rules.
- Keith asked if it might be worth creating a page on the wiki just to track ideas about and pointers to information about rules engines, since it seems increasingly like some form of rules engine is going to play a role in the provisioning space, and took an AI to put up a page with links to the rules engines he's aware of as a starting point.
- Keith then suggested that the discussion is at the top of slippery slope that leads to one or another generalized application and data integration toolkit as a primary component of the effort we're undertaking. He pointed to the Apache ServiceMix toolkit as an example he's quite familiar with from his work with the Bamboo project. Even, he noted, if we're not headed toward the Apache service stack per se, it might be worth looking at to get a sense of what such tools can do for our effort. We might decide to use it directly, or create a constrained subset of the model it implements. Keith agreed to act as "champion" for the ServiceMix toolkit, suggesting that even if ServiceMix as a whole ends up being far more extensive than we need for simple provisioning, having a single vision of such a model is probably better than having no single vision.
- Lucas added that ServiceMix is being evaluated at Berkeley, as well, so it clearly has some community support.
- Rob asked if ServiceMix is entirely Java, entirely C, or something else? Keith noted that while its patterns could be implemented in any language, the existing implementation is all Java.
- **Question #3: Does the scope include any non-registry data sources -- authoritative sources from which target systems may wish to consume data but which may not be considered in-scope by registry providers?**
- Lucas expressed the opinion that the provisioning engine should not be responsible for integrating non-IDM data into target systems scopes, whether it's part of the registry or not... A data item from an ERP may not be something that belongs in the registry, but that doesn't mean the provisioning system (by virtue of having the ability to perform all the necessary operations) should be responsible for getting the data from the ERP and passing it to the target system. There could easily be, he remarked, a non-IDM data integration engine that takes responsibility for that sort of data integration, though, and it might somehow be federated with the IDM data integration engine (eg, the provisioning facility).
- Rob gave as an example of an attribute that might fit in the "not in the registry but useful to target systems" HR/Payroll data that's specific to a position rather than to a person in the position -- Broadbanding status for the position, for example, and asked if the IDM facility should be responsible for making that data available to target systems or if target systems should interact directly with the ERP that sources it.
- Lucas indicated that that would, from his point of view, be out of scope for provisioning.
- Keith added, however, that if we consider provisioning as an instance of information integration, either of the two approaches can work in the higher sense of information integration -- one is simply called "provisioning" and the other isn't. Wherever one systems information has an effect on another system's information, generalized data integration methods can come into play...At this point time ran out for the call, and the call adjourned until the following Wednesday.