SPMLMinutes

SPML Minutes

Agenda

How are SPML gateways being implemented?

How many people are using SPML?

Is SPML viable?

Are there issues with SPMLv2?

Notes

Most of the assembled were familiar with SPML and a handful of participants had already setup or implemented SPML services of some form. Some were using v1 and some were using v2.

UNC started examining SPML because they wanted to implement something that could do whatever Sun Identity Manager could do. Detecting changes, reflecting changes, doing workflow and other necessary processes are functions that would be included in what they may build. No other commercial products that are available met their provisioning/deprovisioning needs, and there are severe time pressures.

This provisioning went into Exchange, calendaring, directories, "anything else we can get our hands on to provision" including research computing resources.

At Yale, they have the sharing of identities, which is largely identity-based. That's distinct from service provisioning. "Provisioning" is a very big term, and most people talk about it in terms of the core netID. But aspects of identity beyond central identity management, or systems that use information beyond identity, are very important to Yale.

Karsten is not sure where you end up drawing the line between user data and service information. Separation of the two seems artificial to him.

Creating an email alias for someone creates an account that is adding an attribute to an identity, while a new email account may be based on a netID or other primary identifier.

In design of a provisioning system, you need to be careful of duplication and overlap. Every case requires an individual analysis. In some cases, the division between these systems can be capricious to Karsten. Yale had hoped that OIM would be that "one" solution, but provisioning of resources is too complicated to do within OIM so it was just much easier to do something else. Karsten wonder if that's OIM's fault as opposed to a fundamental problem. Tabled for further research.

Why do we do this? UNC has identified two major purposes. One is to save duplication in user effort in order to register for campus services. The other part is security; deprovisioning in order to reduce the amount of identity debris and inappropriate access.

UNC will be doing this anyway, and they might want to combine activity if possible. OpenRegistry and Grouper both do bits of this, but not all of it. Other open source projects for provisioning are quiescent. LDAPPC-NG would be useful for some of the provisioning, but it can't do workflow or other functions that are necessary.

A lot of virtual organizations are going to need provisioning and deprovisioning, and LDAPPC-NG will be enhanced to meet those needs. Kuali Rice workflow can also be integrated into Grouper.

Brad thinks the attraction of SPML is that, when changes are being made to a provisioning engine or the endpoint, you have to go through and reperform full testing to make sure both sides are functional. SPML gives you a way to create unit tests which involve simply generation of XML documents for the specific component that has changed. That is the business use of SPML or SAML.

Workflow, to him, is just a queue with people on one side of a process or the other side of a process. A chain of such processes can turn into a full workflow. There can be forks and effective logic gates in the process as well. Workflow is queueing, a queue is a group, and so forth. So the integration of a provisioning engine with Grouper is of some interest to him.

Karsten then described what UNC is doing and building. There's a set of thing that you want to provision, just resources hooked up to services sitting on a service bus. There is an engine in the corner through which all requests are made which decide whether you're eligible or authorized for a service. If you have some data changes, it can decide what synchronizations need to be performed.

A traditional rules engine that is able to do chained rules could do the computation or the logic. Drools is an intriguing choice for that, because it has rules-based eventing and workflow built into it, which obviously is useful for what we're trying to do.

So, if someone comes into the system via a new entry in Peoplesoft, Peoplesoft sends out an XML notice of a new person to the engine. The engine recognizes this is a staff person, and because all staff get Exchange accounts, an Exchange account is created. That's a rule that is associated with the Exchange service listener. Or, if there's an affiliate already in the affiliate system, it could link the new account and send that information off to the directory.

This would all be intermediated by SPML so that the engine could hypothetically be replaced by e.g. OIM.

UNC was interested in contribution of the engine itself, the provisioning service, and had hoped that others would be able to contribute interfaces.

And that engine is very similar to the SPML gateway that Brad is envisioning. These would be essentially OpenSPML libaries, and the SPML messages could then be consumed to actually create events at listeners. The hard part is still writing the interfaces.

Nexus was a project at Memphis that had been started by an old developer who since left to pursue a life of music. It will be replaced later with LDAPPC-NG.

Leif observed that there aren't many services out there that will produce or consume SPML as it stands. The value of a standard is the amount of people who have implemented it already. SPML, to Leif, was one of the standards that was written that never took off. Leif doesn't think we're nowhere near being the egg, chicken, or rooster.

And even the actual systems that say they're ready to handle SPML; some can't provide SPML interfaces but are only willing to dump out XML, and some have shoddy implementations. To Brad, there remains a benefit to having the common language of SPML in the middle, regardless of the support for it at the end points.

Brad also sees a lot of commonality in the systems that we use, so we could contribute provisioning interface implementations between SPML and specific systems that may have poor or nonexistent SPML support. That could, in aggregate, be enough to be meaningfully useful and worth the expended effort. And, it might become compelling to the vendors at some point if enough of this work is done.

John points out that a lot of other vendors, such as Google, have their own fairly qualified API's that can do provisioning well. If he had a central hub that spoke things other than SPML, would that be a good thing? Or do you want to go through SPML first?

SAML as analogy was raised repeatedly, a protocol with knobs all over that can do almost anything through profiling. Good open source and community support for SAML has led to its widespread acceptance by vendors. If SAML hadn't been so cleanly and well profiled, SAML would not be so useful, and Leif thinks getting the semantics right and profiling SPML will be a lot more work. Simply a data representation or messaging format is not enough. The semantics of passing those messages around is, to him, the more important aspect, and writing a library doesn't help that.

Specific semantics for every message have to be decided upon, or there will be only point-to-point solutions. There isn't even necessarily such thing as correct semantics; but there need to be semantics that we can decide upon.

Scott raises the federated provisioning question. To most people, federated provisioning should be the same thing. And to Scott, there are advantages because it forces people to be explicit about a lot of the choices that are made. Sharing internal solutions runs into the whole "my problems are different from your problems" situation, but if this is definitionally a federated solution, then everyone has to agree on what their problems are to some degree.

At some schools, they'll build point-to-point solutions because it's a point-to-point problem to them. A solution will be bought, and in such solutions, it's always going to be easier to hack together code, and where that fails is when you start to federate.

To Scott and John, a generic envelope is not that compelling. The content is going to be application or domain specific, and the fact that the wrapper happens to look similar is not going to save you much code or much effort unless very hard choices and impositions on semantics are made. Otherwise, it's just a big, amorphous bag.

To Scott, SOAP gave way to REST because the envelope was not really interesting to most any systems. Most systems are passing around SOAP payloads and pretending to do REST, and the envelope itself is gone.

Karsten might like to use OpenPTK as an interface, which is an SPI and an API for writing this exact same sort of thing. It's sort-of a gateway framework.

Tom thinks now is the right time to start to address this. The whole "cloud-sourcing" thing that CIO's and CFO's love is one pressing reason, which will involve federated provisioning, for which we are completely unprepared. The Oracle purchase of Sun is another auspicious event. It's a likely workshop topic for the next CSG meeting.

Scott thinks the SPML solution is more sensible from the federated provisioning standpoint, because there are many scenarios that involve multiple identities. But there's a proposal in the SAML committee that's a little weird and doesn't really suit our community well because it provisions information from SP's back into IdP's. But a single identity represented in SAML form might be an appropriate provisioning mechanisms for some applications.

OCLC is developing a cloud-sourced backend office. The large universities are not anticipated clients; community colleges are envisioned. An IdP that will either accept identities from the college that has an IdP, or act as a primary IdP for universities that don't. They'd like to track a variety of information about these individuals that could be batch feeds, or so forth. This is an example use case that we'd like to be able to address with a system.

Deliverables

The short term implementation goals at UNC will be fairly inviolate given how pressing their needs are. A variety of schools were quite interested in seeing the work at its completion, and the work performed seemed very similar to what a lot of other schools were doing.

CAMP and its sponsors are willing to support the work if it wants to keep moving forward. Tom would like to look over the shoulders of the implementers and the deployers. People from CAMP who are interested in this, and those who were unable to attend but interested, are willing to participate.

Does there need to be one group that is generically focused on provisioning, and one on SPML, and perhaps involvement with the standards space? Brad has said SPML specifically, but Tom and others could see the work done separately. SAML will definitionally be always about an individual, not bulk data. But there may be space in here for a SAML-based solution to part of the problem. Conversations have occurred in the past regarding binding of SPML messages to SAML, but it doesn't seem as if those profiles were ever finished.