IdPMetricsDetails

The IdP has extensive logging capability but not much in the way of instrumentation, metrics collection, or real time reporting features. TIER has made some specific preliminary requests for a few pieces of information about deployments for which it would like to enable some kind of collection capability, which may initially be log-based.

The project would like to start collecting requirements from its community on what information deployers, and those who support communities of deployers (i.e. federations), would like to get access to. For the time being, one should assume this might be push or pull, and may include logged, real-time, and/or stored data. Probably all of those will be included over time.

Candidate Data

Feel free to add your requirements to this list. If there's overlap or duplication, we'll dig through it all later, getting it down is the most important thing. We would prefer that somebody adding an item be subscribed to the dev list and able to answer questions about the suggestion in the future, but alternatively just add a contact point for the item to the table in case we need to ask a follow-up.

For consistency, please use the categorization of types of metrics used by the Metrics project: Gauges, Counters, Histograms, Meters, and Timers. Again, exactitude isn't important, it just helps in capturing what is really being asked.

Contact	Item	Type	Currently Available?
TIER	Software Version(s)	Gauge	Java and IdP versions are in the process log and status page but not in a structured way. Nothing else is really captured (container, OS, connected systems).
TIER	Authentication Attempts	Counter	Audit log exposes authentications in terms of requests for profiles that involve authentication (i.e. SSO), and time of authentication could be used to extrapolate whether an actual act of authentication occurred. Failed authentication frequently never exits the IdP or produces any audit trail.
TIER	Attributes Released	Gauge	In audit log on a per-transaction basis. Not clear from TIER requirements whether this is about particular transactions or some kind of inquiry into policy at IdP.
TIER	Attribute filter rules on a per-attribute basis	Gauge	
TIER	Metadata sources list and whether or not signature verification enabled for each	Gauge	Available now in status page but unstructured, except for the signature verification, which is a filter. APIs look like they do provide access to the filters installed, so a status page replacement could do this.
TIER	Heap size (max, usage, available)	Histogram	Available as a gauge in status page, so could be polled and tracked over time.
TIER	MDQ service response time	Timer	
Scott Cantor	Metadata processing time	Timer	Applies to all stages of metadata processing, filtering, etc, and might be interesting to include heap gauge also
Scott Cantor	Request processing time	Timer	Definitely beginning to end, though need to account for the cases where the user's in the middle somehow
Scott Cantor	Request counter	Counter	Should be able to emit counters of various types (requests for various profiles, maybe counters of individual RPs or users or logins or sessions)
Scott Cantor	Expensive operation timings	Timer	Need to assess any places we'd like to gather indications of bottlenecks, certainly would include attribute resolution, scripts, connectors, authentication

Architecture

Logging should continue to be a significant piece of this conversation, because it is obviously not realistic for the IdP to keep a permanent record of all of the things that it does (particularly when there is no requirement for any persistent storage in a lot of deployments to begin with). Any requirements for truly accurate time-series information about requests and things that would normally be audited seem to be better handled through log analysis, but this may include the production of new logging streams, or audit logging formats, that are more suitable for analysis for particular audiences.

On the other hand, the logging framework is already quite exhaustive and we probably need to think more broadly about newer requirements. It probably makes much less sense to add the overhead of logging to capturing performance data, reporting on configuration or environmental characteristics, system health, etc.

We need to be open to a lot of form factors. Both push and pull are probably going to be useful in different contexts. A pull model (e.g., exposing REST /JSON endpoints) is easy to do and probably needed, but creates security complexity in inter-organizational cases. A push model is simpler to enable for a deployer, but requires a little bit of thought around discovery and the possible need for multiple push targets.

In all cases, we should avoid baking much, if any, intelligence about the data being collected so that it will be maximally extensible.

Implementation

We need a framework of some kind and should not need to build all this by hand. At least a couple of candidates have been identified (all Apache-licensed):

- Spring Boot

- Metrics
- JavaMelody

Spring Boot is attractive for newer applications, but it has a number of behaviors that make it a bad fit for the IdP right now, in particular that it's heavily annotation-oriented and very anti-XML Spring configuration, while the IdP is pretty much the opposite. It also provides a lot of its capability through canned features that don't offer much if you have to change how they behave in significant ways, one being the use of Spring Security as the only real security model to protect the automatically exposed metrics. In general, metrics are just a small feature of the overall project.

Metrics seems like a better fit to explore as a basis for a lot of future work, because it plugs into existing code with less fuss, and comes with an extensive framework for collecting different types of data and for reporting it out in different ways. In particular, it should accomodate not only the types of collection we want, but the types of reporting we want, includng the notion of a custom Reporter that would stream or push the information out to a collection point. It should be possible for deployers to configure reporting specific to their needs also, getting us out of the business of maintaining that. One possible caution is whether the use of Spring XML to configure all these features will work, but as long as there are POJOs, that shouldn't be a problem.

JavaMelody is interesting in contrast to Metrics, because it's essentially an outside-in wrapper that can inject itself into a completed application and do measurements of requests, data sources, and other kinds of Java objects it knows about or is taught to understand. It's really more of a deployer's tool, but could be something worth exploring as an installation option if we get more sophisticated with the installer.

Timeline

Ahead of V3.3, which is scheduled for late 2016, none of this exists, and the best options that wouldn't involve invasive new work would be collecting from logs, adding new audit field extractors, and the addition or substitution of a better status page.

If logs can be accessed, then some information may already be available. If the information desired is per-transaction, then it may be available, or may be *accessiible*, meaning that a simple plugin (these can be scripts) could be built that adds an additional field to the audit log, which can then be published through a custom log format that references the field.

If the information is more static and is accessible through walking the web of objects that exist inside the running Spring context, then a simple solution may be to build a JSP page to access the information and report it (probably as JSON). Such a page could be dropped into the IdP web application tree, or could be implemented as a replacement for the existing text-based status reporting page. The page could even be coded to perform a push operation of some kind and scheduled to run with a trivial cron job.

V3.3 will more than likely feature *at least* a better status page as an option or replacement for the current one, and if something were developed by a third party (TIER?) ahead of the release, it could certainly be a candidate for direct inclusion.

It isn't as likely that we will have time to do a lot of additional work on this as part of V3.3, but this is somewhat community-driven. If there's heavy demand, then we'll have to assess, but logout improvements might be a trade-off here.

More extensive work, and certainly the selection of a framework for future work, is probably a given for whatever comes after V3.3, and ideally we would have a framework chosen and incorporated in some minimal way for V3.3 so that third-party work would be possible on top of it.