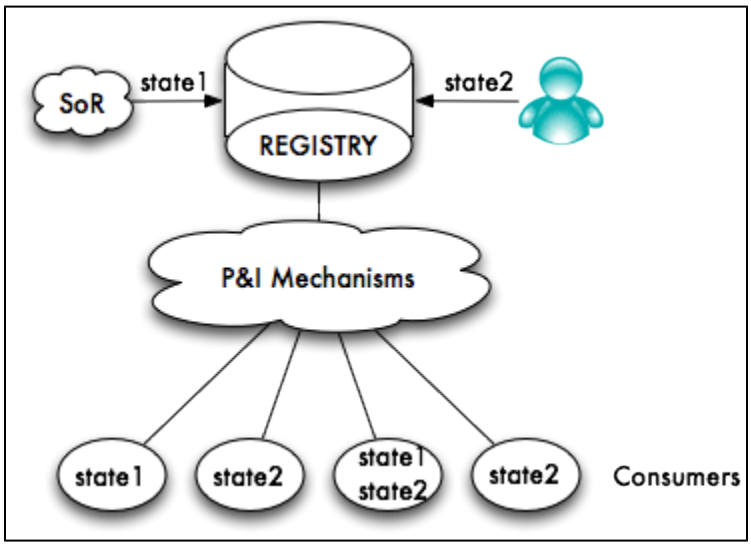
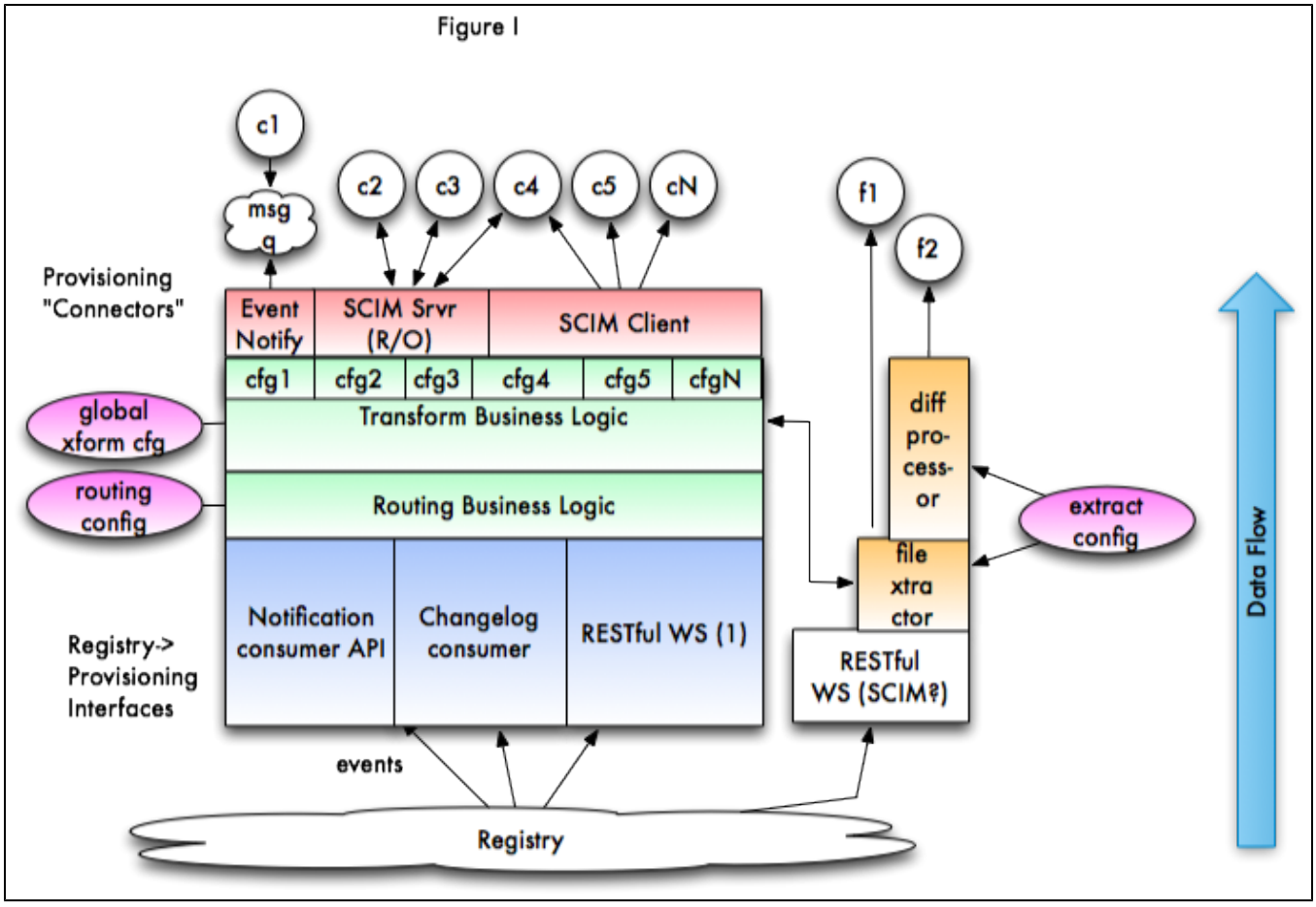
P&I Architecture Diagrams (mark II draft)



The point here is to indicate that our P&I Mechanisms are intended to provide the means by which the state of CIFER resources (typically user-related resources, but not necessarily exclusively user resources) in various consumer systems may be kept consistent (or eventually consistent) with the state of corresponding resources in a CIFER registry. Reconciliation timing, as well as the details of data flows and on-the-wire protocols vary depending on the specific mechanisms implemented and we anticipate that the CIFER P&I suite will need to support a variety of mechanisms in order to interoperate with a range of consumers of varying capabilities. We also anticipate that not all consumers will retain states for all resources in the registry, so selective management of state consistency will be critical.

The following is still quite a work in progress, but as a start at a more detailed depiction of the sort of architecture and capabilities we've been envisioning in our discussions to date (at least with respect to Registry->consumer integration features):



To hit the high points (hopefully):

1. Overall, shaded boxes represent components/capabilities provided by or specified by the P&I suite. Many of these may be implemented using existing technology and delivered recipes, while some may represent new development efforts. Unshaded boxes are expected to be the purview of other workstreams and/or the responsibility of non-CIFER components.

2. Overall data flow is from the bottom of the diagram to the top. Data artifacts reside in the CIFER registry at the bottom of the diagram and they (or events related to changes in their state) are consumed by P&I components, which in turn apply configurable business logic before invoking one of a collection of outward-facing data delivery mechanisms to provide the means for establishing or re-establishing consistency between the registry and various consumers.

3. We anticipate providing two generic "file-based" integration mechanisms for those consumers (labeled "f1" and "f2") that either cannot handle more direct integration or need to be "primed" via some bulk import mechanism. The orange boxes in the diagram depict the two interfaces -- a file extractor interface that relies on a RESTful web service (which we believe could easily be a SCIM interface, and thus leverage some of the code associated with the P&I engine) to collect the current state of relevant objects in the registry, then passes that state information through some configurable transformation logic to produce a "flat file extract" of the current state of the registry in a format suitable for consumption by the requestor. If the requestor requires a full extract, the extractor may deliver it directly, or it may pass the extract through a differential extract generator that either compares the current extract to a previously stored "last" state or compares the extract to a state provided by the requestor to produce a set of differences (adds, modifies, deletes), which can then be returned to the requestor. This implements a consumer-initiated "pull" paradigm that allows consumers to control the rate at which they process state changes, and that can be considered a "least common denominator" for integration patterns, but it's not an integration method we recommend unless other options are infeasible for a given consumer.

4. The large block in the middle of the diagram comprises the components of an event-driven real- or near-real-time provisioning. In the blue layer are the interfaces we anticipate presenting to the registry for acceptance of (essentially, CrUD) events -- a RESTful web service (which may be based on, or perhaps entirely implemented as SCIM) the registry can call to report an event involving a resource in its repository; a polling changelog consumer that can mine changes directly from a registry-defined changelog, and a notification consumer API by which events can be delivered directly through a language-bound API (Java RMI, etc.).
5. The green layer comprises two separate, configurable business logic layers -- one for controlling the routing of events toward specific consumers (implementing both internal access control -- which consumers are allowed to receive what information -- and subscription semantics -- which consumers *need* to receive what information) and one for performing schema transformation (eg., what the registry stores as "preferred first name" may need to be translated to "nickname" for a given consumer). The business logic may be implemented with a variety of tools -- we're strongly considering tools provided by the Shibboleth attribute resolver and attribute mapper for this purpose, but we're also interested in some capabilities provided by the Apache Camel suite and in some other open source BPL interpreters. At this time, we don't anticipate implementing interactive workflow, although if it were necessary for as yet unidentified use cases, this "green" layer would be the logical place for it to reside.

6. The purple ovals represent embedded configuration information that needs to be defined and consumed by the P&I suite, but which we would like to propose as being configured through the CIFER console service. If necessary, P&I should provide a web service interface (RESTful, preferably) for managing configuration objects in these three ovals.

7. The red layer then comprises three key consumer integration modules atop which individual consumer interfaces can be built.

    a. A SCIM client interface. For those consumers that provide a SCIM server interface, the P&I engine will offer a direct SCIM client integration module -- events presented to the engine by the registry and transformed into consumer-specific schema through the business logic layers can be delivered via the SCIM client directly to SCIM-capable consumers. Guidance will need to be provided as to how to maintain schema transformation configurations for SCIM consumer endpoints, but otherwise this should be a "zero-code" integration mechanism, where it's applicable. If there are key use cases identified that would be best served by the SCIM "push" model of integration, P&I may choose to provide example SCIM server endpoints for those consumers, but in general, it's the responsibility of the consumer to provide a SCIM interface for accepting updates.

    b. A SCIM server interface. For consumers that implement "lightweight" event interfaces (eg., that expect to be notified of resources for which events have occurred but expect to then initiate their own retrieval of the resources' states, rather than receiving state information along with the event notification) we anticipate providing a read-only SCIM server interface that provides a retrieval mechanism for the resource information associated with the states for which events occur. [This may be unnecessary, I suppose, if we postulate that the Registry's RESTful web interface is available to these consumers -- they can simply go to that interface for the necessary information, but if we feel we need to restrict access based on routing and leverage transform logic in the provisioning engine, it may be necessary to provide this interface in some fashion.]

    c. A messaging mechanism (likely XMPP and/or using JMS and possibly some queuing system -- ActiveMQ, RabbitMQ, or even Amazon SNS) whereby events filtered and (possibly) transformed through the business logic layer can be posted to one or more message queues, where consumers that prefer to retrieve event notifications via this mechanism can receive such notifications. Delivery of a specific queuing mechanism is out of scope for the P&I effort, although providing standard interfaces for connecting to queuing mechanisms (JMS, XMPP) is in-scope. In some cases, there may be no intervening queuing system, per se, and messages may be delivered directly to consumer interfaces (where that's supported by the consumer).

8. Support for bulk provisioning (both for the inception of new consumers and for resolving state inconsistencies introduced by failed operations during incremental import processes and mishandled events) can be provided via simple flat-file extracts using the file extractor mechanism in orange, or via SCIM bulk provisioning (a full data extract can be processed through the routing and transformation business logic layer and delivered to a consumer as a SCIM bulk provisioning "bundle").