Managing Module Subscriptions I Pipes

In the previous section, we defined a module and outlined a standardized protocol for broadcasting module events. The pattern allows us a to selectively listen for the service events we are interested in.

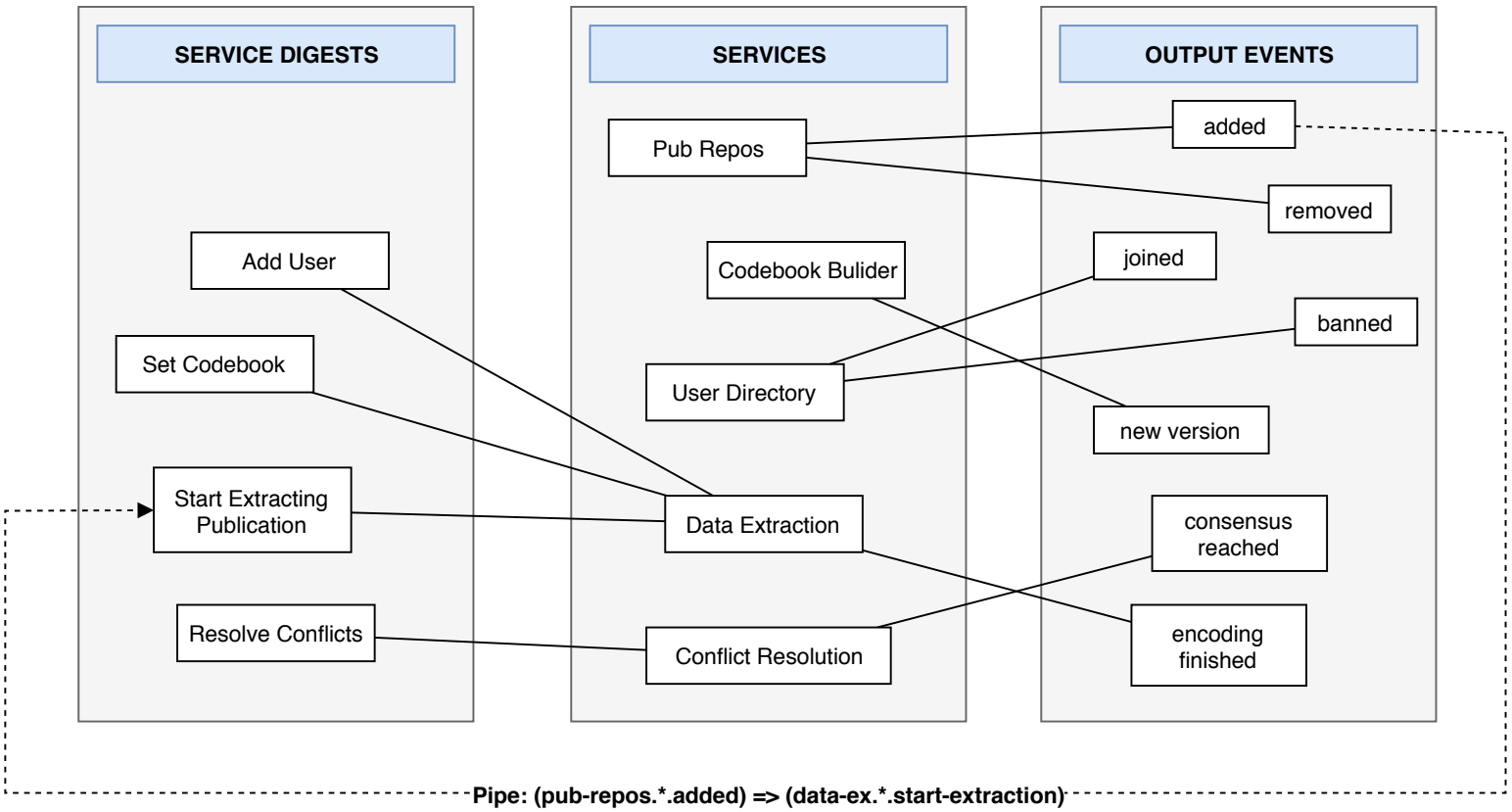
Now we want to introduce a feature that building on this protocol - **module pipes.**

A **pipe** is a mapping of one module's *output* to another module's *input*.

These pipes should be managed in such a way that authorized users can create and destroy them at any time.

The minimum amount of information we need to perform this mapping is:
+ The source service, resource subject, and output event
+ The target service, target resource subject, and digest channel

Notice that if you start connecting "output events" to "service digests", you get a data-flow diagram!
We can do simple validation upon pipe creation to ensure event payloads are compatible



REST API

GET /services/:id/outputs

GET /services/:id/digests

GET /services

GET /services/:id/instances

GET /pipes
- src-output
- src-instance
- tgt-instance
- tgt-digest

POST /pipes
- src-output
- src-instance
- tgt-digest
- tgt-instance
- filters, maps, middleware

DELETE /pipes/:id