Federation

Connecting Livespaces

Federation

- Two levels to federate at
  - Elvin level: federate messages between routers
  - Livespace level: federate entities between rooms

Elvin Federation

- Needed to allow Livespace messages to propagate
- Useful for other Elvin applications such as Sticker

Elvin Federation Model

- Select messages using a subscription expression to be forwarded to another router
- Can either use Elvin's built-in federation, or a federating client like *ewafd*
- Avis 1.0 does not support federation

Federation Expressions

```
   (string (NEWSGROUPS) && string (FROM_NAME) && string (SUBJECT))
|| (string (Group) && string (Message) && string (From))
|| (int32 (Presence-Protocol) || int64 (Presence-Protocol))
|| (int32 (Livespace-Protocol) && Entity-Type != "room")
```

Elvin Federation Config

As before

As before

Master router only

```
federation yes
federation.protocol ewaf://0.0.0.0:2916 # not necessary

federation.class livespace
federation.subscribe livespace \
         (string (TICKERTAPE) && string (TICKERTEXT) && string (USER)) \
      || (string (NEWSGROUPS) && string (FROM_NAME) && string (SUBJECT)) \
      || (string (Group) && string (Message) && string (From)) \
      || (int32 (Presence-Protocol) || int64 (Presence-Protocol)) \
      || (int32 (Livespace-Protocol) && Entity-Type != "room")

federation.provide livespace \ # could also just use TRUE
         (string (TICKERTAPE) && string (TICKERTEXT) && string (USER)) \
      || (string (NEWSGROUPS) && string (FROM_NAME) && string (SUBJECT)) \
      || (string (Group) && string (Message) && string (From)) \
      || (int32 (Presence-Protocol) || int64 (Presence-Protocol)) \
      || (int32 (Livespace-Protocol) && Entity-Type != "room")
federation.link livespace ewaf://SLAVE1:2916
federation.link livespace ewaf://SLAVE2:2916
```

ewafd Approach
(Elvin Wide-Area Federation Daemon)

- A client application that federates messages
- Pro: can use any message filtering logic, modify messages, etc
- Con: lose security credentials
- Con: cannot just federate everything

Federation Effect

- Once federated, can treat routers as a single event space
- Doesn't matter which one you connect to for federated messages
- This is why we can receive global messages in ICS
  - Federate ICS ⇆ DSTO ⇆ Internet

Livespace Federation

- Federation at the entity level
- Assumes Elvin-level federation in place
- Two modes
  - *Mirror*: selected entities of a room appear as if part of another room
  - *Replicate*: synchronise two separate entities so they act as one

Mirroring

- Example: make computer entities in one lab appear in another
- As if two labs had merged

Replication

- Example: synchronise shared clipboards across two labs
- Most useful of the two

Usage

- Run `livespace.services.federation` in one room
- Scenario: allow screen sharing across rooms
- Example configuration:

```
federate.FOCAL<>DJFHQ Battlelab.computer: mirror
federate.FOCAL<>DJFHQ Battlelab.presence: mirror

federate.FOCAL<>DJFHQ Battlelab.teamthink: replicate
federate.FOCAL<>DJFHQ Battlelab.teamscope: replicate
federate.FOCAL<>DJFHQ Battlelab.clipboard: replicate
federate.FOCAL<>DJFHQ Battlelab.screen sharing: replicate
```

Future

- Replication is a good way to connect room services
- Entity mirroring is a hack to work around inability to easily access entities in more than one room at a time
  - Will go away in 2.0