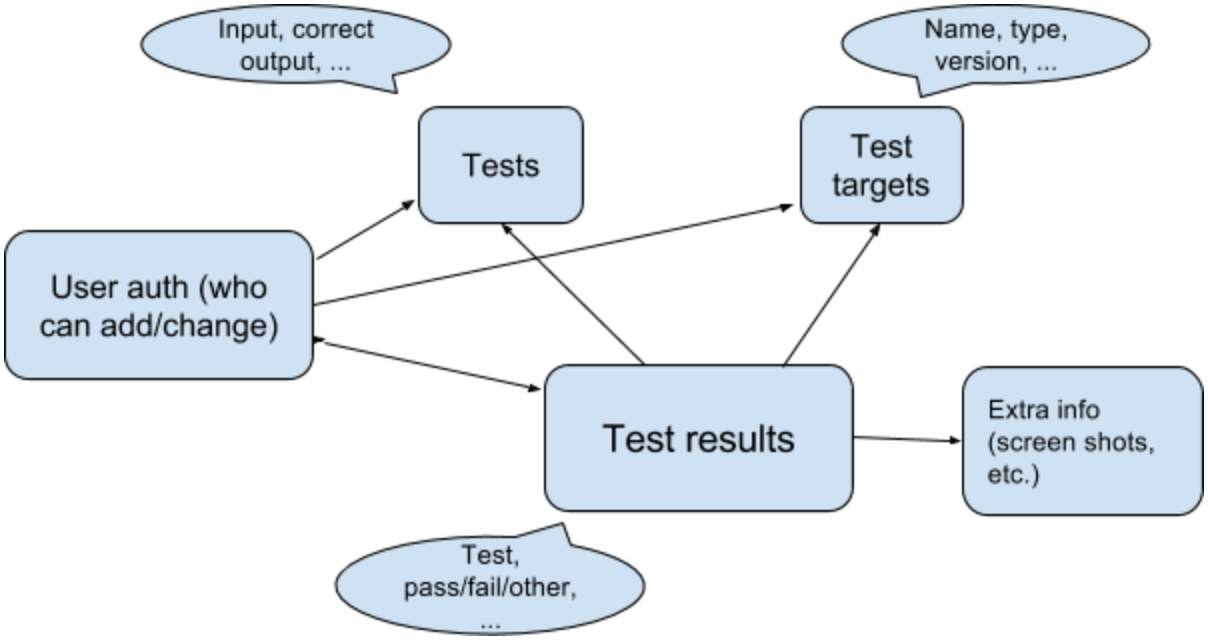
EAI Readiness and Conformance Testing

Overview

There are several parts to this, the first three done more or less in parallel:

- Data management system to track tests and test results
- Library tests, automation for running the tests, and actually running the tests
- Software tests for MTAs, MUAs, and perhaps POP/IMAP servers, automation insofar as it is practical and user tools otherwise, and actually running the tests
- Ongoing maintenance and use

Data management system



The core of this project is a database of tests and results. This serves several purposes. One is enabling reports in multiple dimensions, e.g., what software passed tests A and B, what tests did different versions of software X

fail, and so forth. More important is the ability to add results as new software and versions are available, and to add new tests as we discover new ways that software attempting to to comply with EAI doesn't.

We expect to provide a straightforward web-based interface to the data management system, and also an API so that authorized users can automatically add test results.

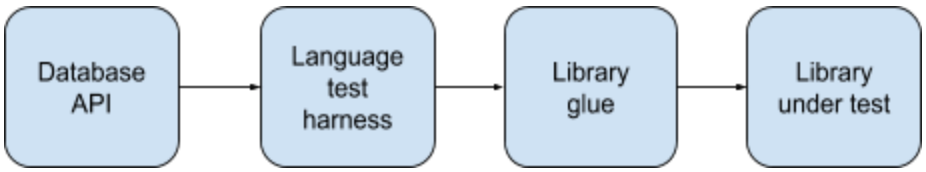
The design and implementation will proceed in parallel with the library and software tests, since experience shows that the initial database design is never quite right.

Deliverables

A. Prototype database design. Initial web and API design and implementation using Django or another mutually agreeable framework.
B. Install and perform library tests, update tools as needed.
C. Install and perform software tests, update tools as needed

Library tests

We believe that it should be straightforward to build automated testing frameworks for most if not all software libraries, so that after the first tests of any particular library, tests of patched or updated versions should involve little more than installing the new library and running the test script.



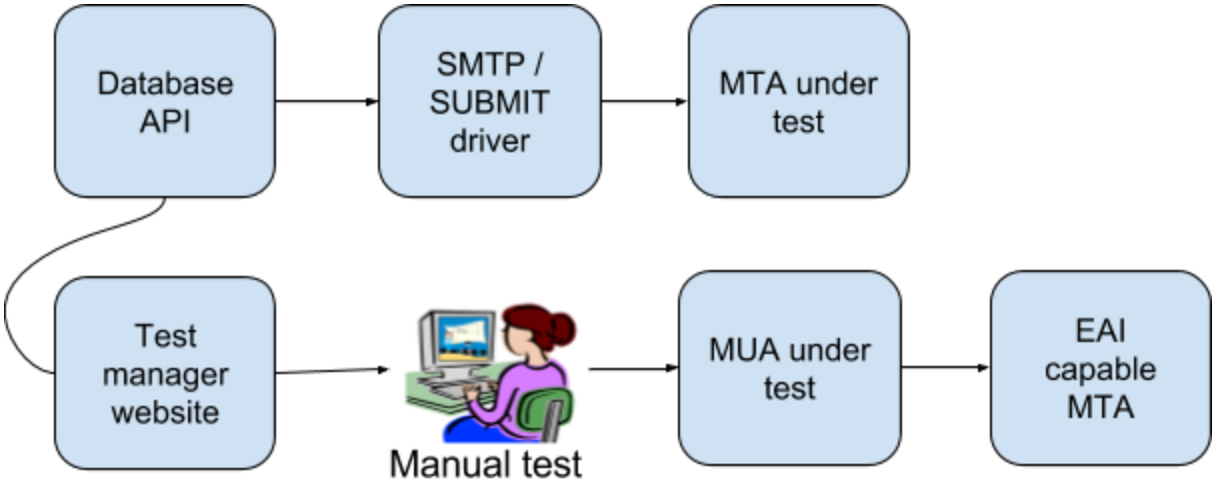
Each programming language in which libraries are written needs a test harness to communicate with the test database, fetch test details, and, if authorized, upload test results. Each library to be tested needs "glue" code that deals with the specific names of library routines and adapts to the details of data formats for the particular library.

Deliverables

A. Identify libraries to be tested, and the required test harnesses.
B. Define all the tests.
C. Build one test harness and one or two glue packages,. Run tests.
D. Build the test of the test harnesses and the test of the gleue packages. Run the rest of the tests.

Software tests

Testing software is conceptually similar to testing libraries. For MTA tests, a test driver connects to the MTA over a network and runs the tests, sending commands to the MTA and checking the results. For MUA tests, it is unlikely to be practical to script all of the different mail programs and webmail systems, so a human user runs the tests, guided by instructions on the test manager web site and entering test results and, when useful, MUA screenshots, into the test manager. The MUA will have to be configured to communicate with an MTA that is known to implement EAI.



Deliverables

A. Identify MTAs and MUAs or webmail systems to test.
B. Build SMTP/SUBMIT driver, test one MTA.
C. Test the rest of the MTA.
D. Build test manager website. Test one MUA.
E. Find testers, test other MUAs.

Ongoing maintenance and use

After the agreed tests are complete there is likely to be ongoing maintenance, such as:

- Train people to run tests
- Train people to develop new library glue for new libraries

- Add new tests as new areas of potential incompatibility are discovered
- Manage users who can run tests and upload results
- Improved analytics, such as graphics and spreadsheet export