fhiso

CFPS 6
(Call for Papers Submission number 6)

Partial Implementation and Extensions

Submitted by: Tychonievich, Luther

Type: Functional requirements

Created: 2013-03-15

Last updated: 2013-04-16

URL: Most recent version: http://fhiso.org/files/cfp/cfps6.pdf
This version: http://fhiso.org/files/cfp/cfps6_v1-1.pdf

Description: Round-trip communication between tools supporting different portions of the data standard should behave sanely. Nine use cases demonstrate what that means.

Keywords: partial implementation, extension, collaboration

Partial Implementation and Extensions

Luther A. Tychonievich

12 March 2013

Abstract: *Tools should be able to support subsets of any standardized data model and to add additional extensions to it. Two such tools should be able to communicate without harming one another's data.*

I accept as self-evident that tool developers will want to include data that is not part of an standardised data model. I would also prefer to allow tools to implement just a part of a data standard if they wish. Sharing data between two tools supporting different elements of the data standard with extensions should not compromise the integrity of either tool's data. However, tools should not be required to transport unknown payloads inserted by other tools' extensions.

In the following nine use cases, I represent data conceptually as a list of symbols. For tools A and B, a_i and b_i represent custom extensions to the data model or standard elements not handled by the other tool; x_i represents information that is part of the standard data model and handled by both tools.

Addition Round-Trip

B should be able to add data to what A sent it, and A should see that addition.

1. A sends B the data (x_1, a_1)
   B adds to that (x_2) and returns (x_1, x_2) to A
   A should now see (x_1, x_2, a_1).

2. A sends B the data (x_1, a_1)
   B adds to that (x_2, b_1) and returns (x_1, x_2, b_1) to A
   A should now see (x_1, x_2, a_1).

Removal Round-Trip

B should be able to remove from what A sent it, and A should see that removal.

3. A sends B the data (x_1, x_2, a_1)
   B removes x_2 and returns (x_1)
   A should now see (x_1, a_1).

4. A sends B the data (x_1, x_2, a_1)
   B removes x_2, adds b_1, and returns (x_1, b_1) to A
   A should now see (x_1, a_1).

Replacement Round-Trip

A combination of the two cases above: B should be able to change that A sent it, and A should see that change.

5. A sends B the data (x_1, x_2, a_1)
   B replaces x_2 with x_3 and returns (x_1, x_3) to A
   A should now see (x_1, x_3, a_1).

6. A sends B the data (x_1, x_2, a_1)
   B replaces x_2 with x_3, adds b_1, and returns (x_1, x_3, b_1) to A
   A should now see (x_1, x_3, a_1).

Custom Edit Round-Trip

Tools should be able to handle other tools' extensions and successfully communicate that to one another.

7. Suppose A sends B the data (x_1, a_1)
   B understands a_1 and adds to it a_2, returning (x_1, a_1, a_2) to A
   A should now see (x_1, a_1, a_2).

8. Suppose A sends B the data (x_1, a_1)
   B understands a_1 and removes it, returning (x_1) to A
   A should now see (x_1).

9. A sends B the data (x_1, a_1).
   B understands a_1 and replaces it with a_2, returning (x_1, a_2) to A
   A should now see (x_1, a_2).

   Observe that this case suggests that B must communicate to A in some fashion that it removed a_1 rather than simply not including it.