SAMP Web Profile

Mark Taylor (Bristol)

Jonathan Fay (Microsoft Research)

(work funded by Microsoft Research)

IVOA Interop
Nara, Japan

8 December 2010

`$Id: websamp.tex,v 1.19 2010/12/08 05:40:20 mbt Exp $`

Outline

- The Problem
- Proposed Solution
- Pros and Cons
- Next Steps?

Target Capability

- SAMP works well for *desktop clients*
- Would like it to work for *web clients* (code running in a browser)
  - In-browser technologies:
    - JavaScript (a.k.a. JScript, ECMAScript)
    - Java applet/WebStart
    - Flash
    - Silverlight
  - Example capabilities:
    - Provide a button which sends a table/image/spectrum to a suitable desktop viewer (many potential pages)
    - Receive information from desktop clients, e.g. highlight catalogue rows (e.g. Ivan Zolotukhin's Open Clusters Catalog)
    - Communicate with other web pages loaded in the same browser (e.g. Andrew Conolly's ASCOT, Alyssa Goodman's "Seamless Astronomy")

The Problem

- Standard Profile communications require:
  - Discover Hub:
    - ▷ Locate user's home directory ~
    - ▷ Read `~/.samp` file
  - Make calls to Hub:
    - ▷ POST to HTTP server on localhost
  - Receive callbacks from Hub:
    - ▷ Run HTTP server
  - Dereference data URLs:
    - ▷ GET from http/ftp/file URL

The Problem

- Standard Profile communications require:
  - Discover Hub:
    - 🛑 Locate user's home directory ~ — user ID/dir not available
    - 🛑 Read `~/.samp` file — local file I/O not allowed
  - Make calls to Hub:
    - 🛑 POST to HTTP server on localhost — cross-domain HTTP not allowed
  - Receive callbacks from Hub:
    - 🛑 Run HTTP server — many in-browser environments can't run servers
  - Dereference data URLs:
    - 🛑 GET from http/ftp/file URL — cross-domain HTTP etc not allowed
- Security restrictions imposed by browser "sandbox"

Browser “Sandbox”

Purpose

- Restrictions imposed

  . . . *by* the browser

  . . . *on* the web-based client code

  . . . *on behalf of* the user
- Prevents web-based code from executing with user privileges
- Result is that visiting a web page is not as dangerous as installing an application

Restrictions:

- Local filesystem I/O
  - ▷ Web client cannot access local filesystem
  - ▷ How to get round it?
    - ◦ Escape the sandbox (run outside of browser-imposed restrictions)
- Cross-Domain blocking
  - ▷ Web client can’t do HTTP access except to the server that it originated from
  - ▷ How to get round it?
    - ◦ Escape the sandbox (run outside of browser-imposed restrictions)
    - ◦ Use some *cross-domain workaround(s)*

Possible Solutions

Possible solutions, as discussed at previous meetings:

- Signed Java Applet
    - ▷ Runs outside sandbox with user confirmation
    - ▷ WebSampConnector (VO Paris Data Centre)
    - ▷ Needs java
    - ▷ Problems on some browsers? (not sure about this)
- Browser Plugin
    - ▷ Runs outside sandbox when installed by user
    - ▷ Sébastien Derriere's PLASTIC/SAMP Firefox plugins
    - ▷ Very browser-specific
- Alternative Profile
    - ▷ Uses cross-domain workarounds, avoids local file I/O
    - ▷ read on . . .

Alternative Profile

- Alternative profiles explicitly permitted in SAMP
    - SAMP = generic core + specific profile(s)
    - Profile = hub discovery + RPC encoding/transport + callback arrangements
    - Currently (SAMP v1.1/1.2), only Standard Profile defined
    - Door left open for other possibilities
- Web Profile:
    - Need something that will allow a sandboxed application to find and communicate with hub

Proposed Web Profile Details

Like Standard Profile (uses XML-RPC), but:

- Hub Discovery:
  - ▷ Hub server resides on well-known port (`http://localhost:21012/`)
- Hub Communications:
  - ▷ Hub XML-RPC HTTP server uses all known *cross-domain workarounds*
  - ▷ These are configured to allow *maximum accessibility* from all sandboxed clients
- Callbacks:
  - ▷ Reverse HTTP/"Long poll" pattern
    - Client pulls callback instructions from hub, rather than hub pushing to client
    - Client may make repeated periodic short-timeout polls,
      or blocking long-timeout requests
    - Hub response contains XML-RPC (`<methodName>`, `<params>`) pairs
- Data URL Dereferencing:
  - ▷ Hub provides proxy service for external URLs

Cross-Domain Workarounds

- Cross-domain access from within the browser sandbox
  - Common requirement (Flickr, Twitter, YouTube, Amazon, . . .)
  - HTTP server somehow declares sandboxed clients may access its resources
  - Several client- and browser-specific options exist:
    - ▷ Implement Cross-Origin Resource Sharing standard
      - Server reads/writes HTTP headers to signal cross-domain policy to browser
      - W3C standard (`http://www.w3.org/cors/`)
      - JavaScript support in XMLHttpRequest Level 2 (Firefox 3.5+, Chrome 2.0+, Safari 4.0+)
      - JScript support in XDomainRequest (IE8+)
    - ▷ Serve `/crossdomain.xml` resource
      - Server provides XML file(s) describing cross-domain policy to browser
      - Introduced by Adobe Flash
      - Flash support since version 7(?)
      - MS Silverlight support in all(?) versions
      - Java support for (unsigned) applets and JNLP in versions 1.6.0_10+
    - ▷ Serve `/clientaccesspolicy.xml` resource
      - Works like `crossdomain.xml`
      - MS Silverlight support (preferred alternative to `crossdomain.xml`)

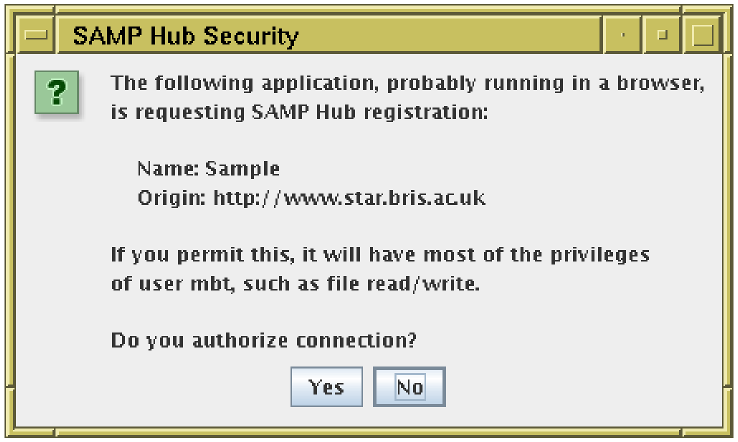
Implementation

- Hub
  - Implemented using JSAMP
- Clients
  - JavaScript SAMP client *(tested)*
    - ▷ Different strategies required for different browsers:
      - ◦ Use XMLHttpRequest Level 2 if present
        (Firefox 3.5+, Chrome 2.0+, Safari 4.0+)
      - ◦ Else use XDomainRequest if present (Microsoft)
        (IE 8+)
      - ◦ Else use ugly hack which mimics cross-domain XMLHttpRequest using Flash behind the scenes (flXHR)
        (anything with Flash plugin)
    - ▷ JavaScript client library can hide the details
      (`http://www.star.bris.ac.uk/~mbt/websamp/`)
  - Flash SAMP client *(kind of tested)*
    - ▷ Just works
  - Silverlight SAMP client *(almost tested)*
    - ▷ Should just work
  - Unsigned Java applet/JNLP SAMP client *(so far, not working)*
    - ▷ Should just work, but only for browser Java plugin 1.6.0_10+

Security ???!!!?

Security

- Security not *HTTP level*, but *applied at Registration time*
  - HTTP access alone to hub server can do nothing dangerous
  - Only registered clients have privileged access (private key acquired at registration)
  - User must supply explicit permission when client requests registration


SAMP Hub Security

The following application, probably running in a browser, is requesting SAMP Hub registration:

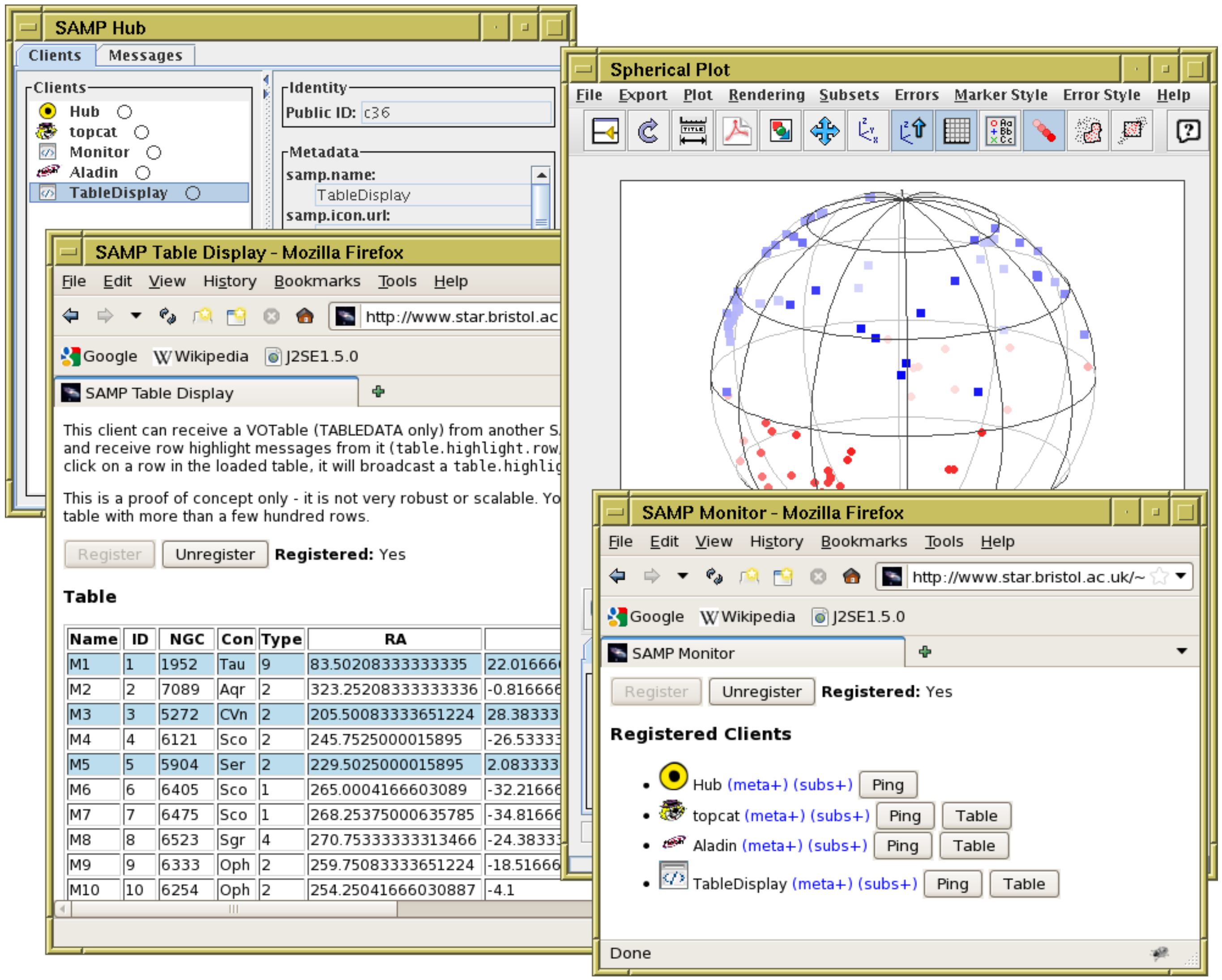
Name: Sample
Origin: http://www.star.bris.ac.uk

If you permit this, it will have most of the privileges of user mbt, such as file read/write.

Do you authorize connection?

- Similar level of security to signed applets
  - User has to click OK to grant privileges
  - Technology (signed certificates) for signed applets is much more sophisticated . . .
  - . . . but details ignored by 99% of users

Demo



Clients
- Hub
- topcat
- Monitor
- Aladin
- TableDisplay

Identity

Public ID: c36

Metadata

samp.name:
TableDisplay

samp.icon.url:

This client can receive a VOTable (TABLEDATA only) from another S
and receive row highlight messages from it (table.highlight.row
click on a row in the loaded table, it will broadcast a table.highlig

This is a proof of concept only - it is not very robust or scalable. Yo
table with more than a few hundred rows.

Registered: Yes

Table

Name	ID	NGC	Con	Type	RA	
M1	1	1952	Tau	9	83.50208333333335	22.01666
M2	2	7089	Aqr	2	323.25208333333336	-0.816666
M3	3	5272	CVn	2	205.50083333651224	28.38333
M4	4	6121	Sco	2	245.7525000015895	-26.53333
M5	5	5904	Ser	2	229.5025000015895	2.083333
M6	6	6405	Sco	1	265.0004166603089	-32.21666
M7	7	6475	Sco	1	268.25375000635785	-34.81666
M8	8	6523	Sgr	4	270.75333333313466	-24.38333
M9	9	6333	Oph	2	259.75083333651224	-18.51666
M10	10	6254	Oph	2	254.25041666630887	-4.1

Registered: Yes

Registered Clients

- Hub (meta+) (subs+) Ping
- topcat (meta+) (subs+) Ping Table
- Aladin (meta+) (subs+) Ping Table
- TableDisplay (meta+) (subs+) Ping Table

http://www.star.bris.ac.uk/~mbt/websamp/

Pros and Cons

- ✓ Web-based SAMP clients possible without Java
    - ▷ JavaScript, Flash, Silverlight, others?
- ✓ Low overhead for web-based SAMP clients
    - ✗ all-browser JavaScript solution is still messy
    - ✓ but matters should improve as more modern browsers prevail
- ✗ Fragments SAMP client base
    - ✓ Mitigated if all known hubs implement both profiles
- ✗ Increases burden on hub implementors
    - ✓ JSAMP in hand
    - ✓ SAMPy — Luigi willing in principle
- ✗ Security issues
    - ✓ Comparable with signed applets
- ? Facilitates move of applications off desktop into browser

Where Next?

- Do we agree Web Profile is a good idea?
  - Otherwise, stick with WebSAMPConnector (Java) or no web clients
- If so:
  - Standard Document:
    - ▷ New section of SAMP standard or separate Recommendation Track doc?
    - ▷ Push forward or wait for implementations?
    - ▷ First draft attempt available on the web (HTML, PDF)
  - Hub implementations:
    - ▷ JSAMP dual-profile hub release
    - ▷ SAMPy dual-profile hub development?
  - Client implementations:
    - ▷ JSAMP web profile client library implemented
    - ▷ JavaScript client library implemented (but not well)
    - ▷ Other client-side library development? (Flash, Silverlight, . . .)
    - ▷ Experimental/production SAMP web client applications?

Discussion?