PLASTIC 0.5
PLASTIC 0.5
PLASTIC design goals
PLASTIC design goals

• Simplicity
• Easy for client developers - at the expense of infrastructure developers
• Pragmatism - go with what works most of the time
Overview of PLASTIC
Overview of PLASTIC

Client Application
Overview of PLASTIC

Client Application

Hub
Overview of PLASTIC

register

“My name is Topcat. I understand messages: foo, bar, donkey”
Overview of PLASTIC
Overview of PLASTIC
Overview of PLASTIC

“an app has registered”

“My name is Aladin. I understand message ‘load table’”
Overview of PLASTIC
Overview of PLASTIC

“load table”
Overview of PLASTIC

• Hub permits synchronous and “asynchronous” messaging
• Broadcast to all, or point to point
• Hub filters on message type
Overview of PLASTIC
Overview of PLASTIC

"Aladin has unregistered"
Transports:
Transports:
Transports:

xml-rpc

Java-rmi

?
What’s a message?
What’s a message?

- A Plain Old URI (typically an ivorn)
- A bunch of parameters

ivo://votech.org/votable/loadFromUrl
(id, url)
What’s a message?

- A Plain Old URI (typically an ivorn)
- A bunch of parameters
- The URI is opaque and defines the syntax and the semantics (to a greater or lesser extent)
- Message types developed ad-hoc by devs
What’s a message?
What’s a message?

• Messages vary in type and specificity
  • Specific command: “show this area of sky”
  • General command: “here’s a bunch of data points, do something”
  • Event: “The hub is about to shut down”
• No formal taxonomy
PLASTIC=>SAMP
PLASTIC=>SAMP

http://www.ivoa.net/twiki/bin/view/IVOA/PlasticOnePointOh
12 changes
“API” & infrastructure
Refactor “API”
Refactor “API”

• Necessary for the other changes

• Specifically:
  • Separate the registration and metadata declaration phases
App

Now:

Hub
Now: 

register(name, messages)
Now:

register(name, messages)

id

“app registered”
Proposal:

ALADIN is an interactive software sky atlas developed by the CDS, allowing one to visualize digitized images of any part of the sky, to superimpose entries from astronomical catalogs, and to interactively access related data and information.

WARNING: THIS VERSION IS A PROTOTYPE RELEASE. FEATURES HAVE BEEN ADDED FOR DEMONSTRATION PURPOSES. USE IT AT YOUR OWN RISK.

Or take the official version for a real usage: http://aladin.u-strasbg.fr.
Proposal:

register()

id

“app registered”
Proposal:

`register()`

`id`

`declareMetaData(name etc)`

“metadata changed”
Refactor API
Refactor API

- **Pros**
  - Allows metadata to change
  - Conceptually “cleaner”

- **Cons**
  - Makes registration two step
Generalise Metadata
Currently:
Currently:

- Register with:
  - Name: topcat
  - Supported Messages: ivo://votec.org/...
Currently:

- Register with:
  - Name: topcat
  - Supported Messages: ivo://votech.org/...
  - Respond via messages for
    - name, description, logo, ivorn

ivo://votech.org/info/getIconURL
Current system
Current system

• Pros:
  • Extendable via new messages

• Cons:
  • Split of registration/message metadata
  • Inefficient
  • Can’t be supplied by “unmessagable” apps
Proposal
Proposal

• Declare all metadata post-registration, store in Hub

ivoa.name=Topcat
ivoa.description=TOol for Processing...
ivoa.logoUrl=http://..../tc3.gif
Proposal

• Declare all metadata post-registration, store in Hub

• Metadata can be updated at will

• IVOA-defined keys for “common” metadata, but extendable ad-hoc

  ivoa.name=Topcat
  ivoa.description=TOol for Processing...
  ivoa.logoUrl=http://..../tc3.gif
  ivoa.logo16X16Url=http://...../smaller.gif
Proposal

- Declare all metadata post-registration, store in Hub
- Metadata can be updated at will
- IVOA-defined keys for “common” metadata, but extendable ad-hoc

```
ivoa.name=Topcat
ivoa.description=TOol for Processing...
iraf.worker=true
```
Proposal

- Declare all metadata post-registration, store in Hub
- Metadata can be updated at will
- IVOA-defined keys for “common” metadata, but extendable ad-hoc
- All metadata optional (clients must fall back gracefully).
Generalise metadata

- Include supported message type metadata, or declare separately?

```
ivoa.name=My App
ivoa.mtypes=foo, bar, donkey
```
Message IDs (new)
Message IDs (new)

- The hub will assign a session-unique ID to each “request” message
- The ID will be reused by “reply” messages
- Essential to pair up requests and responses under asynchronous mode (cf JSON-RPC)
Bootstrap method
Bootstrap method

- Now:
  - Read a file in a “well-known” location

```plaintext
#PLASTIC server uk.ac.starlink.plastic.ServerSet
#Sun May 13 23:35:52 BST 2007
plastic.xmlrpc.url=http://john-taylors-computer.local:2112/
uk.ac.starlink.plastic.servid=uk.ac.starlink.plastic.ServerSet@7a6686
plastic.version=0.4
plastic.rmi.port=1099
```
Bootstrap method

- Now:
  - Read a file in a “well-known” location
- Pros:
  - Simple, widely accessible
- Cons:
  - Inelegant, unreliable?
Security
Security

- Proposal is to add a basic level of security without complicating the protocol

- Currently vulnerable to:
  - Port scans
  - App spoofing
Now:
Now:
Now:

port scan
sendMessage(gaia_id, "exec", "rm -r *")

Risk reduced by using a firewall, but still vulnerable on multiuser machines
sendMessage(gaia_id, "exec", "rm -r *")

Risk reduced by using a firewall, but still vulnerable on multiuser machines
sendMessage(gaia_id, "exec", "rm -r *")

Risk reduced by using a firewall, but still vulnerable on multiuser machines
Proposal:
Proposal:
Proposal:

register()
Proposal:

- register()
- register(secret)
Proposal:
Proposal:

```
sendMessage(sender_id, destination_id, ...)
```
Proposal:

private id

aladin: public_id

sendMessage(sender_id, destination_id,...)
Proposal:

private id

aladin: public_id

sendMessage(sender_id, destination_id,...)
sendMessage(sender_priv_id, destination_pub_id,...)
Proposal:

aladin: public_id

sendMessage(sender_id, destination_id,...)
sendMessage(sender_priv_id, destination_pub_id,...)

private
id
Security
Security

• Do we allow anonymous access?
Synch vs Asynch
Synch vs Asynch
Currently:

Synch
Currently:

Synch request
Currently:

Synch response
Currently:

Asynch
Currently:

Asynch request
Currently:

Asynch response (discarded by hub)
Synch vs Asynch?

- Stay as we are
- Use of message ids to tie together asynch request & response messages
- Drop synch messaging altogether
Messages
Message param typing
Message param typing

- Now:
  - Messages are typed using a subset of XML-RPC types

<table>
<thead>
<tr>
<th>Tag</th>
<th>Type</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;i4&gt; or &lt;int&gt;</td>
<td>four-byte signed integer</td>
<td>-12</td>
</tr>
<tr>
<td>&lt;boolean&gt;</td>
<td>0 (false) or 1 (true)</td>
<td>1</td>
</tr>
<tr>
<td>&lt;string&gt;</td>
<td>string</td>
<td>hello world</td>
</tr>
<tr>
<td>&lt;double&gt;</td>
<td>double-precision signed floating point number</td>
<td>-12.214</td>
</tr>
<tr>
<td>&lt;dateTime.iso8601&gt;</td>
<td>date/time</td>
<td>19980717T14:08:55</td>
</tr>
<tr>
<td>&lt;base64&gt;</td>
<td>base64-encoded binary</td>
<td>eW91cGxhbjdid0IHZJcWQgdC</td>
</tr>
</tbody>
</table>
Message param typing

• Now:
  • Messages are typed using a subset of XML-RPC types
  • Proposal: everything a string, array, struct
  • Why? What do we gain from typing?
Named Message Params

- Now: parameters identified by position
- Proposal: parameters identified by name
  - Pros: Easier to make messages extendable and parameters optional
  - Greater clarity
mtypes vs ivorns
mtypes vs ivorns

• Why IVORNS?
  • Dereferencable URIs
  • Provide a human (and machine?)-readable definition of a message type
  • Searchable: find me an application that understands this message type
mtypes vs ivorns

• Why IVORNS?
  - Dereferencable URIs
  - Provide a human (and machine?)-readable definition
  - Searchable: find me an application that understands this message type

No message IVORN has ever been registered
Message annotations
Message annotations

- Annotations allow the recipients of messages to annotate mtypes in order to narrow down the semantics but do not change the syntax of a message
Message annotations

• Annotations allow the recipients of messages to annotate mtypes in order to narrow down the semantics but do not change the syntax of a message

• A hypothetical example:

  process.votable.url ➔ You will receive a URL
  It points to a VOTable
  The params are: id, url
Message annotations

- A receiving client declares that it understands
  - process.votable.url@load
  - process.votable.url@overwrite
- A sending client does not understand load, overwrite - they’re merely used as labels for the user of the client.
Example
Message Annotations

• Does not preclude more specialised types such as `display.votable.url` that map to the same functionality

• Pros:
  • Flexible, extendable, dynamic
Distinguish refs
Distinguish refs

Dropped
Transport
Recap:
Recap:

xml-rpc

Java-rmi

?
Recap:

xml-rpc

Java-rmi

?
Transport options:

• Exactly one protocol

• XML-RPC?

• One mandatory protocol + optional ones?

• All protocols optional?
Let battle commence...
Let battle commence...