Volume Title
*ASP Conference Series, Vol. **Volume Number***
Author
*© **Copyright Year** Astronomical Society of the Pacific*

SAMP App Launcher - An on-demand VO application starter by JMMC

Sylvain LAFRASSE[1], Laurent BOURGES[1], Guillaume MELLA[1]

[1]*UJF-Grenoble 1 / CNRS-INSU, Institut de Planetologie et d'Astrophysique de Grenoble (IPAG) UMR 5274, Grenoble, F-38041, France*

Abstract. *SAMP*[1] is the dedicated Virtual Observatory protocol to ensure data exchange between compatible astronomical software running on personnal computers. However, one *SAMP* weakness lies in its requirement to have interoperable applications already running in order to gracefully ensure communication between them. To circumvent this requirement, we present a dedicated application, plus some new *SAMP* specifications, focused on Java^{TM}software available through the *Java Web Start* application-deployment technology (*JNLP*)[2] at this stage. *JMMC*[3] *AppLauncher*[4] software fakes any described application by registering stub clients on the central *SAMP* hub. When one of the fake clients is solicited by any third-party software, *AppLauncher* takes the responsibility to start the true application, and then forwards the waiting *SAMP* message once fully started. To achieve this, we propose a set of new *SAMP* key-value pair to hold *JNLP* URLs. In the future, other kind of software packages technology could also be supported. We also want to standardize this solution, and get one central registry-like interoperable repository of compatible software, in order to open our mechanism to any third-party *SAMP* application provider. To illustrate, we briefly present our own use case, which demonstrates the need of such a tool for the JMMC applications suite.

1. Problem : How to Bootstrap our Tool Suite ?

The *Jean-Marie Mariotti Center* provides software to astronomers involved in optical interferometry technics, to help them use the most powerful observation facilities in the domain. We offer three main Java applications, namely *ASPRO2*, *LITpro* and *SearchCal*, to support scientists from observation preparation phase up to data analysis.

Those applications recently gained interoperability capabilities. They can now exchange data and provide services to each others (and all other VO-compliant software, such as *Aladin* or *TOPCAT*). To achieve this, we rely on *SAMP* - the dedicated VO protocol for inter-application communication - and more specifically on *jSAMP*, its reference implementation in Java.

[1]Simple Application Messaging Protocol - `http://www.ivoa.net/samp/`

[2]`http://download.oracle.com/javase/6/docs/technotes/guides/javaws/`

[3]Jean-Marie Mariotti Center (a.k.a as JMMC) - `http://www.jmmc.fr`

[4]`http://www.jmmc.fr/applauncher`

But *SAMP* interoperability cannot work until a hub is started and all scientific applications run, gently waiting for astronomers actions. Which means end users should already be conscious of all this to interoperate our software. Then come the chicken-and-egg problem - how the user could easily interoperate our software even if he is not conscious of this capability, or does not remember where to find our applications for launch? What can we do to enhance our user-friendliness, *SAMP*-wise ?

2. Solution : Faking, then Launching SAMP Clients !

A first idea was to dedicate one application to start the three others at once. But this was far from elegant, resource and compatibility-wise.

The idea made its way, to later become a *SAMP* capable application launcher. The first thing our application should do is hosting the *SAMP* hub outside of any other scientific software. This way the hub is always running, even if the user quits all but our application (which will be prevented by explicitly informing the user of the consequences of killing the *SAMP* hub for interoperability). And at long last, our application should be able to download and start any scientific software needed by the user, at runtime. *AppLauncher* was born !

AppLauncher workflow is roughly synthesized hereafter:

- On startup, *AppLauncher* sets up a *SAMP* hub (or hooks to any hub already available), then registers as many fake *SAMP* clients as it internally has application metadata descriptions for;
- Once a fake client is solicited by the user through any *SAMP* message, *AppLauncher* traps and holds this message for differed delivery;
- *AppLauncher* then tries to start the corresponding real application using *Java Web Start* technology, handling errors and timeout as far as possible;
- Once the third-party application is fully started and registered in *SAMP*, *AppLauncher* unregisters the fake client from the hub, and then forwards the waiting message to its true scientific recipient for processing.

2.1. Faking SAMP Clients

Fake clients are synthesized at runtime by *AppLauncher*. They pretend to be their real application by presenting the same name and the same *SAMP* capabilities to the hub. But their sole purpose is to forward whatever message they receive to the true application, like a proxy. This is great because nothing has to be changed for this to work, no *SAMP* hub hack nor third-party applications modification required !

With our 'fake client' mechanism, the end user is able to virtually access and send any *SAMP* message to any application not even running on his computer, the only limit being the number of fully described application metadata available to *AppLauncher*. We could also apply the same 'fake client' method for *MIME* type associations with file extensions, to better integrate applications capable of handling certain file formats to modern operating systems for example.

Of course not all *SAMP* messages should be eligible for such automatic application startup. For example, *SAMP* broadcasting (that target a large number of applications) or `table.highlight.row`, `table.select.rowList`, `coord.pointAt.sky` messages (that potentially occurs many times in a raw at high frequency) should not trigger automatic applications startup.

2.2. Launching SAMP Clients

To begin with this complex issue (in regard of all the different executing platforms available out there), we decided to first rely on *JNLP* software distribution mechanism. This gives us several advantages to launch scientific applications:

- *Java Web Start* is platform-agnostic, and works well on the three main desktop operating systems that are Mac OS X, Linux and Windows, freeing us of lots of trickeries and portability issues.
- As long as an Internet connection is available, *JNLP* will always present users with an up-to-date version of the missing application.
- If Internet is not available, the latest downloaded version (cached in *Java Web Start* system) will be used instead.

JNLP main drawback (even if VO applications are often written in Java) lies in its restriction to handle Java packages only. To circumvent this, we also envision a way to use Java's ability to execute certain script languages such as Python or JavaScript. Third-party scientific application providers could then develop their own scripts to properly handle download, install and execution of non-Java software in a platform-agnostic way. HTTP URL could also easily be opened in the user's default web browser to handle *SAMP*-compatible web applications.

3. Standardization : Opening to Third-Party Apps

AppLauncher first release embed fake clients for:

- *ASPRO2*, *LITpro* and *SearchCal* - the three *JMMC* Java applications;
- *Aladin*, the CDS reference sky atlas;
- *TOPCAT*, the reference interactive graphical viewer and editor for tabular data.

But our ultimate goal is to make *AppLauncher* work for any *SAMP* software out there. So the crucial system that is missing today for full integration of third-party applications is a centralized repository providing application metadata, such as:

- a mandatory application name, to present the user with a rightly named fake client (also required as a strict match is needed to internally link fake and true applications for hub connection management);
- a mandatory URL pointing to a *JNLP* (and maybe later other means to describe any application download and startup process, such as scripts), to easily start the application at runtime;
- a mandatory icon (standardized to 64*64 PNG);

- a list of supported *SAMP* messages (a.k.a `MTypes`[5]), should they be public or private;
- a list of handled *MIME* types for file opening (with corresponding icons if any);
- at least an URL, pointing to application's main web page.

So until such a central registry become widely available, JMMC will host its own temporary solution to centralize those essential metadata. And to easily fill our registry, we will embed a *SAMP* connection sniffer to discover user's third-party apps (of course only if he acknowledges to contribute). That's why we need at least one new keyword now in *SAMP* specifications to point each application's *JNLP* URL.

4. Perspectives

In the near future, a first round of beta-testing will be held on IVOA `apps-samp` mailing list to gather as much feedback as possible from software experts. The next logical step will then be an (hopefully open-source) public release of *AppLauncher*.

Meanwhile, there is an ongoing effort to update *SAMP* specifications with new keywords to support our needs. Further discussions are also going on through IVOA mailing lists about application metadata registry, in order to define a standard that fits everybody's needs.

We are also trying to shape a broader way to download and start non-*JNLP* applications, maybe by relying on Rhino or Jython to execute 'download and start' scripts that could be written by application providers.

We are actively working on an XML schema describing application metadata, so that developer can provide us with their own description for integration in *AppLauncher* !

Acknowledgments. The JMMC would like to thank:

- Mark Taylor (University of Bristol, Astrogtrid), main maintainer of:
  - *SAMP* Virtual Observatory standard, available at `http://www.ivoa.net/Documents/latest/SAMP.html`;
  - *jSAMP*, *SAMP*'s reference implementation, available at `http://software.astrogrid.org/doc/jsamp/`;
  - *TOPCAT*, the reference interactive graphical viewer and editor for tabular data, available at `http://www.starlink.ac.uk/topcat/`.
- Centre de Donnees Astronomiques de Strasbourg (CDS), for the reference sky atlas *Aladin*, availbale at `http://aladin.u-strasbg.fr/`.
- *jMCS*, the underlying GUI framework of *AppLauncher* and other scientific Java applications such as *ASPRO2*, *LITpro* or *SearchCal*, available at `http://www.jmmc.fr/dev/jmcs`.

[5]`http://www.ivoa.net/cgi-bin/twiki/bin/view/IVOA/SampMTypes`