Astronomical Data Analysis Software and Systems XXI
ASP Conference Series, Vol. 461
Pascal Ballester, Daniel Egret, and Nuria P. F. Lorente, eds.

SAMP: Application Messaging for Desktop and Web Applications

M. B. Taylor,[1] T. Boch,[2] J. Fay,[3] M. Fitzpatrick,[4] and L. Paioro[5]

[1]*H. H. Wills Physics Laboratory, University of Bristol, U.K.*

[2]*CDS, Observatoire astronomique de Strasbourg, France*

[3]*Microsoft Research, One Microsoft Way, Redmond WA, U.S.A.*

[4]*National Optical Astronomy Observatory, Tucson AZ, U.S.A.*

[5]*National Institute for Astrophysics, IASF, Milano, Italy*

Abstract. SAMP, the Simple Application Messaging Protocol, is a technology which allows tools to communicate. It is deployed in a number of desktop astronomy applications including ds9, Aladin, TOPCAT, World Wide Telescope and numerous others, and makes it straightforward for a user to treat a selection of these tools as a loosely-integrated suite, combining the most powerful features of each. It has been widely used within Virtual Observatory contexts, but is equally suitable for non-VO use.

Enabling SAMP communication from web-based content has long been desirable. An obvious use case is arranging for a click on a web page link to deliver an image, table or spectrum to a desktop viewer, but more sophisticated two-way interaction with rich internet applications would also be possible. Use from the web however presents some problems related to browser sandboxing. We explain how the SAMP Web Profile, introduced in version 1.3 of the SAMP protocol, addresses these issues, and discuss the resulting security implications.

1. Introduction

Astronomers as a group use many different software tools to deal with different kinds of data. A typical science analysis might use separate specialised applications to manipulate catalogues, images and spectra, and exchanging data or coordinating activity between these can complicate the workflow and impede interactivity. A single monolithic application providing all possible capabilities for all astronomical data types might be an attractive idea, but it is clearly not a practical prospect. However, by enabling the existing set of specialised tools to interoperate seamlessly, something approaching the utility of a do-everything tool can be achieved with realistic levels of effort.

With this in mind, SAMP, the Simple Application Messaging Protocol, has been developed within the International Virtual Observatory Alliance. First standardised in 2009, it is a direct descendant of PLASTIC (Boch et al. 2006), the Platform for Astronomy Tool InterConnection, developed earlier within the European VOTech framework. Its origin within the Virtual Observatory (VO) movement has been stimulated by use cases arising from VO capabilities and fostered by the collaborative spirit within the community of VO developers, but it is equally suitable for use in non-VO scenarios. It has been deployed to date in many popular desktop astronomy tools.

2. Architecture

The architecture of SAMP is divided into three distinct layers:

Abstract API: Defines the structure of messages and responses, the available data types, and the services provided by the Hub.

Profile: Prescribes how the concepts defined in the Abstract API are mapped to actual communication operations, such as bytes on a wire, using a particular transport protocol. In particular prescribes how a client can locate the Hub.

MTypes: An open-ended list of semantically distinguished message types, labelled by short, hierarchically structured, strings. An MType resembles the specification of a subroutine in an API, and defines the semantics of the message as well as zero or more required or optional named parameters and return values.

The Abstract API and Profile are deliberately separated, in order to allow different choices of Profile to support different operating requirements. In early versions of SAMP, only a single profile, the Standard Profile, was defined. In the Standard Profile the transport protocol is XML-RPC (a simple HTTP-based RPC mechanism), and Hub discovery is via a "lockfile" in the user's home directory.

The MTypes are in most cases defined outside of the standard by mutual agreement between client developers. Useful tool interoperability is achieved when an MType produced by one tool is consumed by another; a simple and commonly used example is "`image.load.fits`" which permits one tool to send an image in FITS format to other tools which know how to do something with such an item. Any client developer may introduce a new MType; it becomes useful to the extent that other interoperating clients introduce support for it.

SAMP operates using a star-like topology, in which a "Hub" provides directory and message-brokering services to clients. All direct SAMP communication is therefore client–Hub; the Hub forwards messages and responses appropriately to achieve client–client messaging. In order to participate in SAMP communications, clients must first *register* with this Hub. Once registered, clients can participate in publish-subscribe type messaging, in which each client may declare its willingness to receive messages with certain MTypes. The Hub is a daemon process, which is conceptually distinct from any of the clients, though it may in practice run within one of the clients. This is in fact commonly the case; many SAMP-aware applications will launch a Hub on startup if one is not already running, and this means that users typically run SAMP whenever they use their favourite analysis tools, without taking any explicit action to do so.

3. Design Principles

The aim of SAMP is to deliver maximum interoperability between astronomy tools *in practice*. Clearly, this requires a protocol which is sufficiently capable, efficient and expressive to be able to exchange useful data and control information between applications. However, it also requires that it becomes widely deployed into the tools that astronomers use, and that once deployed, astronomers actually discover and make use of the capabilities. The latter part of this requirement is at least as difficult to achieve as the former, and so the design of SAMP has given particular consideration to making it easy to adopt and use.

To address one aspect of this, the design has focussed on making it as easy as possible for application developers to implement SAMP functionality in their software. The standard is independent of language and operating system, and as few assumptions as possible are made about the implementation environment. Some language-specific SAMP libraries exist, but even in their absence a programmer can implement without much effort a basic function such as transmitting an image or table to other tools, as long as HTTP and XML libraries are available. Where design conflicts have arisen between ease of implementation or use, and rigour or reliability, the former has taken precendence.

A second property of SAMP usage is that the semantics of the messages are typically rather vague. The "`table.load.votable`" MType simply says "here is a VOTable" and it's up to the receiving tool to display the table cells, or plot row positions on a sky image, or calculate statistics on it, or do whatever is appropriate in the context of the receiver. The fact that there are a few, widely used, MTypes in common usage means that by supporting a few generic operations appropriate to its capabilities, a given tool has a good chance of interoperating usefully with other tools likely to be running in the same environment, without needing to know what they are. This model does not lend itself to providing detailed external control of a tool; however any tool may additionally provide such an interface via tool-specific MTypes.

Finally, where possible, the design has been extensible. The general rule is that provision is made to permit optional, non-standard behaviour as long as it does not compromise the capabilities of clients which implement only the standard.

The result of all this is that users can run whatever selection of desktop astronomy tools they choose, and there is a good chance that they will find these tools can operate together as a loosely integrated suite. The burden on tool developers is modest; in particular it is not necessary for the developer of one tool to have a detailed understanding of the operation or capabilities of any of the others, or to make assumptions about what other clients will be registered at run time.

4. SAMP for the Web

A huge amount of content and functionality is available to astronomers from the World Wide Web, and since SAMP's inception it has been clear that allowing web, as well as desktop, applications to communicate using SAMP would be highly desirable. The designation "web application" covers any code running within a web browser, from a few lines of styling JavaScript to Rich Internet Applications like the web version of Microsoft's World Wide Telescope. The most important browser-based execution environments in this context are at time of writing JavaScript, Java, Adobe Flash, and Microsoft Silverlight. Of these JavaScript is much the most prevalent — effectively all browsers support it, and a large proportion of web pages use it to some extent.

Unfortunately, there are some technical obstacles to implementing SAMP access for web applications. These are a result of security restrictions imposed by web browsers: browsers run downloaded code in a "sandbox" from within which access to sensitive resources on the local host is deliberately blocked. The most significant restriction is that sandboxed applications are not allowed access to the local filesystem or to external ("cross-origin") URLs. This precaution is necessary to prevent accidental or malicious damage to the local system by web pages the user visits. In order to ex-

tend SAMP communications to web applications therefore, some workaround for this sandboxing is required.

Of the browser platforms mentioned above, the only one which provides such a workaround controllable from the web application itself is Java. If a Java applet is *signed*, the browser permits it to access local resources. This mechanism has been used by the VO Paris Data Centre team as the basis of a component named WebSampConnector, which has been used successfully by a number of web applications. However code signing can be expensive or unsatisfactory, and it requires the use of Java which is often not convenient for web developers.

Version 1.3 of the SAMP standard (Taylor et al. 2012) provides a more general solution by defining and implementing a new Profile, which allows access from sandboxed web applications, alongside the existing Standard Profile described in Section 2. This Web Profile differs from the Standard Profile in four main respects: the Hub is contacted on a well-known port; the HTTP server on which the Hub resides implements one or more cross-origin workarounds which signal to the browser that web applications should be permitted to access it; the Hub provides a proxying service for cross-origin URLs; and client callbacks are achieved using a "long poll" mechanism.

This solution delivers easy access to SAMP communications from JavaScript and other in-browser environments. What are the security implications? Given this access to the Hub, a potentially hostile web application can request SAMP registration. If registration is denied, it can take no further action. If accepted, it can perform two potentially sensitive operations: exchange SAMP messages, and read cross-domain URLs. A security conscious Web Profile implementation can therefore take the following measures: first, always solicit explicit authorization from the user at registration time, by means of a popup dialogue. Second, restrict the messages the client may send to a known list of harmless MTypes. Third, restrict the URL proxying service so that the client cannot access local resources for which it has no legitimate need. With these measures in place, a hostile web application can affect the local system only if the user explicitly allows it (presumably if misled by some phishing-like attack), and even in that case, the worst damage that can be done is fairly harmless, for instance sending unwanted FITS images to an image viewer.

5. Conclusions

By designing for interoperability in practice, SAMP has achieved integration of many desktop tools for astronomers. Version 1.3 of the protocol extends this to data and functionality delivered on the World Wide Web. Incorporation of this new capability into the protocol has been facilitated by the clean division in the standard between the abstract API and its mapping to communication operations.

Acknowledgments. MT gratefully acknowledges support for work on this project from the UK Science and Technology Facilities Council and from Microsoft Research.

References

Boch, T., Comparato, M., Taylor, J., Taylor, M., & Winstanley, N. 2006, PLASTIC - A Protocol for Desktop Application Interoperability, Note, IVOA

Taylor, M., Boch, T., Fitzpatrick, M., Allan, A., Paioro, L., Taylor, J., & Fay, J. 2012, SAMP, IVOA Recommendation 1.3, IVOA. arXiv:1110.0528