

# Could Jiro Extend the Jini Pattern Language?

Position Paper for the OOPSLA 2000 Workshop on the Jini Pattern Language

[Anthony.Earl@sun.com](mailto:Anthony.Earl@sun.com)

1<sup>st</sup> September 2000

## Introduction

In this short note I wish to explain why it is appropriate to include a discussion of Jiro™ in any work on the Jini pattern language. There are several reasons behind this and they include:

1. Jiro is implemented with the aid of some Jini services;
2. Jiro exposes some Jini services and extends others;
3. Jiro and Jini originate from the same corporation and are both Java™ technologies;
4. Jiro has only just been released and potential users need guidance on the most appropriate use of the technology in their problem domain;
5. The Jiro manufacturers would appreciate investigation and input from the patterns community on improving the usefulness of the product to better meet the needs of their customers.

In the remaining space I will briefly describe what Jiro is and the kinds of problems it tries to address. I will discuss some of the relationships between Jini and Jiro. And then I will go on to describe some examples of how patterns for Jiro would be extensions of the Jini pattern language and how valuable patterns for Jiro itself would be.

## What is Jiro?

Jiro is a newly-released Java technology from Sun Microsystems Inc. It is an industrial-strength compliant implementation of the Federated Management Architecture (FMA)[FMASpec] that emerged from consensus in the storage industry through the Java Community Process. Since space here is so limited I will describe what the FMA attempts to provide through a problem description and through analogy with another Java technology, Enterprise Java Beans.

The use of electronic storage is growing at a tremendous rate and the E-business explosion probably means that further acceleration is ahead. Those system administrators responsible for storage (sometimes called, "Storage Administrators") are finding serious difficulties in keeping pace with growing demand for their skills and services. The lack of storage management person-hours is becoming a bottleneck in business processes and expansion. The subsequent lack of expert planning is creating a growing

headache for those who are finding temporary solutions in just wheeling in more servers. More efficient solutions such as Storage Area Networks (SANs) require more expertise and the support of management tools.

The tools to help storage administrators are often specific to the hardware vendors and this means that in heterogeneous shops the storage administrators are faced with collections of inconsistent tools to manage their devices one at a time. The FMA defines a 3-tier architecture and component model for storage management systems that can be composed of components from multiple vendors. Those tiers are the device-tier, the middle-tier, and the presentation-tier. Only the middle-tier is specified in the FMA standard. This allows a variety of device-specific instrumentation technologies to be used to discover, monitor, and control devices and whatever choice of user-interface that the storage management integrators finds appropriate for their solutions. However the middle-tier has a Java-based, defined component model and interfaces to services that are guaranteed to be available in any FMA implementation.

Such an approach has proven dramatically successful in the E-business world given the rapid adoption of Enterprise Java Beans(EJBs). Only the middle-tier interfaces are defined in the specification yet this architecture has been supported by a wide range of compliant implementations and users. It is important to understand why EJB technology is not particularly suitable in the storage domain. In the E-business domain there are typically just a few databases or database tables that are accessed by potentially millions of end-users. The database structure and presence are relatively stable over long periods of time. The transactions that the databases support are typically very short and can be relatively easily rolled back into a consistent state. These kinds of requirements are supported well by EJBs.

In the storage management world there are typically less than a handful of users (storage administrators). They manage thousands of devices. The set of devices is relatively unstable. New ones are regularly coming online while existing ones break down or are put off-line for maintenance. A typical transaction may involve moving data from one set of devices to another and only doing that after a safe backup to tape. This can take hours and, since we are dealing with real-world entities rather than the logic held in database tables, there are many actions that cannot be rolled back (e.g. formatting a partition). There is still a need to have long transactions reach some kind of consistent state even if failure occurs. These kinds of requirements are not well supported by EJBs. Middleware with different attributes and services is required. That is where the FMA and its implementation in Jiro fits in.

## How are Jiro and Jini Related?

The FMA specifies a set of services that Jiro offers including: lookup; event; security; transaction; controller; persistence; logging; and scheduling. Some of these are defined as Jini services but the standard has no requirement that Jini itself be the method of implementation (just as RMI is often used in Jini solutions but is not a requirement). Examples of this would be the lookup and transaction services.

Some of the Jiro services are extensions of Jini services. An example of that would be the Jiro Event service. In Jini, just as in the Java event model, the subscriber for an event becomes strongly coupled with the sender of the event. In the Jiro event model, the Event service plays an intermediary role so that subscribers only have to find the Event service and can remain unaware of the source of the events. The notions of observing listeners and responsible listeners is also added.

Jiro reuses Jini constructs such as leasing and defining interfaces with Java. Thus the good dynamic system properties that Jini offers can be preserved. In fact, Jiro services are all valid Jini services.

Some characteristics of Jiro are deliberately extensions of the Java programming language. Examples here include the ability to call remote static methods and remote constructors (which is not possible through RMI or Jini). More sophisticated examples include the use of Jiro's Context mechanism for security, transactions and controllers.

Some characteristics of Jiro are specific to satisfying the requirements of being a real-world component model. Examples of these include the deployment of jar files to Jiro Stations; the installation of Jiro components through invoking remote constructors and the facilities for defining persistent components that will automatically restart across Station shutdowns .

There are some aspects of Jiro that specifically address the needs of the storage community. No new language is used to define interfaces (just use Java) which means device management facades are particularly easy to start with for those new to objects and Java. And there's a general design principle for Jiro that the programming and invocation of distributed components should look as much as possible just like writing local code.

## Areas of Pattern-Work for Jini & Jiro

Here are some ideas of where patterns for Jini and Jiro would help developers and users better understand those technologies and how to effectively use them.

1. Specifying the patterns for existing Jini and Jiro services;
2. Capturing how Jini is used to implement Jiro services;
3. Patterns for proposed new Jini or Jiro services/components;
4. Patterns for the use of compositions of Jiro services;
5. Patterns that are used in the implementation of Jiro (e.g. the use of an extended Java Virtual Machine (JVM) for Jiro Stations and the use of Referents and Acceptors as extensions of the skeleton and stub approach of RMI);
6. Reaching an initial state of a Jiro-based storage management system (e.g. what are the different ways that device management facades can be usefully started?);
7. Handling the introduction of new hardware into a Jini or Jiro domain in the real world.

## Conclusion

I'd just like to finish by mentioning that Jini pattern that extends into the Jiro arena would have a substantial audience from the numerous supporters of Jiro that are listed on the Jiro web site[Jiro web site].

## About the Author

Anthony Earl has a B.Sc. and a Ph.D. from the University of York, UK. He was the author of the ECMA Reference Model for Software Engineering Environments and a book on the same subject while working at HP Labs in Bristol, UK. He has worked for the Software Engineering Institute in Pittsburgh PA. He is currently a Jiro Consultant for Sun Microsystems, Boulder, CO.

## References

[FMASpec] Federated Management Architecture (FMA) Specification, Version 1.0, Sun Microsystems Inc.

[Jiro web site] <http://www.jiro.com>