Introduction to the proposed Apache Metro Project

This document presents an overview of the code, project relationships, status and general roadmap of the suite of sub-system that will collectively make up the initial code repository together with an introduction to the community and our establishment strategy.

Product Breakdown

Product	Description	
Metro	The metro product (formerly known as Merlin) is a component management system backed by a formal component type and block management system capable of support composite component deployment. *Primary subsystems include:*	
	Transit	The transit system is a resource gateway that provides functional support for the creation of classloader hierarchies based on remotely available artifacts. The transit system serves as the bootstrap deployment environment for the metro runtime and magic build system.
	Logging	The logging sub-system the handles the establishment of pluggable logging implementations including Lo4J and LogKit.
	Meta	The component type model dealing with the declaration of runtime requirements towards a composition system.
	Composition	The component management model (context, configuration, channels, dependencies management), that provides the framework for the composition and the interface to the underlying deployment runtime.
	Activation	The activation platform is a plugin established by the metro kernel that encapsulates the runtime contract of a particular component model. It is responsible for the control over lifecycle and lifestyle aspects.
Studio	The Studio product is an Eclipse plugin that provides support for development of composite components through management of meta information about component types and meta data about service composition.	
Magic	Magic is an ant library that provides support for centralized version management, transitive dependency management, and a suite of common build functions related to composite component development processes.	

Technical Strategy and Priorities

The metro platform is characterized by strong contracts, from meta-information collocated with component classes though to deployment information packaged under units called blocks. Through explicit separation of the publication of component operations requirements from deployment solutions, the metro team has established an architecture within which it is possible to compose new component implementations dynamically on demand. This notion of composition is a strong and important characteristic fundamental to the delivery of component reuse.

A second and notable aspect of the technical strategy is a strong separation of api, spi, and implementation units across all aspects of the metro platform. This notion of strong separation is reflected thought our development tools, the runtime platform, and the deployment infrastructure.

Looking forward, the team aims to deliver solutions supporting long-running systems maintenance, dynamic sub-systems replacement, graceful platform evolution, and enhancements dealing the integration of peer systems. This last aspect presents probably the most interesting social aspect the metro platform – the ability to enhance the development and runtime process of connected peers. From this notion is an opportunity to strongly reinforce and amplify the value to and generated by the end-user community.

The challenges ahead will cover many technical domains including, security, distribution, and availability management (and non-availability tolerance). Achieving these targets will require the continued process of building the developer community, continued support for end-users, and the engagement of the private sector in areas concerning support, training, and related professional services.

Related Apache Projects

Project	Relationship
Ant	The metro build system is build directly on the Apache Ant build system and leverages many of the new features introduced in the recent 1.6 release cycle.
Avalon	The metro platform provides and will continue to support the Avalon 4.2 framework contract as part of its concurrent model management strategy.
Jakarta	Many of the commons utilities are used with the metro sub-systems, including cli, collections, beanutils, and commons logging. Commons dbcp and pool are used with metro facilities related to database connection management and the commons messenger and digester are used in facilities supporting message integration. In addition, the metro platform leverages the Jakarta regular expressions and becel utilities. The Jasper compiler and runtime library from Jakarta Tomcat is used within the metro http facility.
Logging Services	Development in metro has lead to the establishment of implementation independent logging apis and spis capable of supporting plugin providers. Discussions have been initiated on the subject of collaboration with the Apache Logging Service Project with a view towards contributing to solutions in the logging service area that facilitate improved contract and service provider management.

Community

The development community surrounding the proposed Metro project is predominately made up of developers with experience in enterprise applications delivery and typically a background in dealing with problems of reuse and long-term maintenance concerns.

The end-user community is made up of a very diverse collection of individuals representing domain activities in the financial services sector, information systems in the bio-technology sector, embedded applications in the area of transaction and cash management, larger scale applications in the area of payment processing, applications in the business object and workflow area, instant messaging systems, and a variety of desktop applications.

Establishment Strategy

The Metro project codebase is based on a fork of the Apache Avalon project codebase. The proposed fork would involve the division of the current Avalon codebase in such a way that would separate existing Avalon Merlin product, sub-systems and facilities, from the framework, LogKit, and cornerstone component content. A critical consideration within this process is the absolute maintenance of code compatibility with the released framework 4.2 component contracts, while at the same time, enabling the concurrent management of alternatives.

A brief summary of the remaining package is presented in the following table.

Package	Issue/Responsibilities
Framework	The Avalon Framework current release is version 4.2. An improved version 4.3 has been established under svn and could be moved to release status with minimum effort (no api changes). It is important that the framework receive basic maintenance as it is used across many projects in Apache and has significant exposure outside of Apache. Based on discussion within the Avalon development community - further development of 4.X is not anticipated.
LogKit	The LogKit product is a stable logging system. It not under active development as it has already a good level of maturity. It receives occasional maintenance. The package is referenced by the framework (an issue addressed and resolved under the non-released framework 4.3), it is used in the Excalibur Logging system, and it serves as one the underlying technology for one of the plugin logging implements under within the Merlin platform. LogKit requires a home against which on-going but light weight support can be provided. The Excalibur project may be a candidate.
Cornerstone	The Cornerstone project is a small collection of components based mainly on the Excalibur components (cornerstone representing carse-grain service functionality whereas the underlying Excalibur components are more fine-grain almost utility level systems). Cornerstone components are used in a number of projects (notably the Apache James Project). Further development of the suite is not anticipated – but minimal maintenance should be provided.

References to maintenance mainly concern the long term monitoring of Gump builds. This is already in place however resolution of successful build cycles for a number of cornerstone packages is pending resolution of technical issues related to Gump/Maven integration. It is expected that these issues will be resolved in the near term following which normal cycles will resume.

Notes concerning Migration

Concerns related to package changes and migration overhead will depend directly on the type of user and usage scenarios. The following is a summary of the requirements on the Metro team and the probable impact on users.

Usage Role	Impact	Assessment
Component Author	Zero	The existing release of the Merlin 3.3.0 platform will be maintained under the Metro project. A release of Metro will supercede Merlin but shall maintain full backward compatibility with the Avalon framework 4.2 api contract – including standard context entry usage and lifecycle extension support. Looking further ahead the Metro team aim to prove support for concurrent runtime systems, enabling the possibility of the simultaneous deployment of framework 4.2 based components with a native metro equivalent.
Facility Developers	Minor	Facility developers will be exposed to package name changes at the level of the composition system api.
Embedded Applications	Moderate	Changes to package names would be visible and in addition some changes are anticipated with respect to the management of the repository system initial context. These changes reflect some important enhancements to the initial bootstrapping framework that will in the longer term provided higher resilience and greater flexibility in the management and upgrading up large installation. Impact will likely require plugin descriptor re-generation and potential changes to code dealing with the kernel establishment.