METRO Roadmap

Mission Statement

The Metro project aims to provide advanced enterprise class solutions for component based development, dynamic deployment and runtime management.

Overview

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

Key features of the metro runtime platform include the following:

* automatic assembly
* muli-layer configuration management
* advanced context management
* composite component management
* packaged deployment scenarios
* local and remote repository integration
* classloader and plugin management
* integral extension jar management
* logging service framework and selectable plugin implementation
* complete facilities architecture
* pluggable runtime layer
* lifecycle lifestyle management

In addition to the above – the metro project provides developer tools supporting tight integration of version and dependency management, pluggable build extensions, complete build automation, testing, packaging and continuous integration solutions

Product Breakdown

Product	Description	
Metro	The Metro product (formerly known as Avalon 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.

Product	Description	
	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 operational 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 solution – the ability to enhance the development and runtime processes of connected peers (customers, providers, partners, etc.). 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, continuing our support for end-users, and the engagement of the private sector in areas concerning support, training, promotion, and related professional services.

Related Apache Projects

Project	Relationship
Ant	The metro build system is built 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 will continue to provide complete support for the Avalon 4.2 framework contract as part of its concurrent model management strategy. The Metro project will a replacement framework optimized for the metro build and runtime as a concurrent upgrade 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 are used within the metro http facility.
Logging Services	An IOC logging api, service provider management spi together with a Log4J and LogKit implementation plugins. The Metro project intends to expolore the possibility of collaboration with the Apache Logging Service project with respect the API and SPI layers.

Community

The development community surrounding the Metro project is made up of the core developer team. These individuals have strong 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 will be derived from a fork of the Apache Avalon codebase. The project will maintain a maintenance branch corresponding to a current Merlin 3.3.0 platform and all related Avalon sub-systems. A production branch will be established under the metro namespace.

Following establishment of the codebase and initial infrastructure work will be undertaken to rationalize implementation systems with the objective of establish tight integration between transit, the magic build system, and the metro deployment platform. Artifact distributing via DPML will be maintained as is.

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 projected impact on users relative to the primary API and SPI contracts.

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. The Metro team aims to rapidly deliver 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 and context entry URI 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 important enhancements to the initial bootstrapping framework that provided higher resilience and greater flexibility in the management and upgrading up large installation.