Draft: Integration Coordination

A manual for Departmental Software Platforms and Services (DSPS)

Infrastructure Software Development and Architecture (ISDA), IS&T, MIT

Document Summary

This is a manual. It describes policies and recommendations regarding the management of application platforms by the Departmental Software Platforms and Services team (DSPS). This will include the management of all assets required to reproduce and run server applications, such as source code, configuration, and procedures for release to production support.

This document is used to instruct work and for audit guidelines.

Revision History

08/25/08	Steve Landry	Source-Control policies
08/29/08	Steve Landry	Additions about managing acess-control lists
09/07/08	Steve Landry	Changes to installation procedures. Added Pipeline process information.

Table of Contents

Process and Control

Pipeline Planning

ISDA processes change requests through the ISDA Pipeline Planning process. ISDA management chooses the members of this group. All tasks related to new systems implementations, enhancement, or problem resolution come through this group.

Ticketing

DSPS uses the Request Tracker queue ISDA::Admin to track all change. Team members should only add tickets for work orders once they know the work is approved by Pipeline.

Several pieces of information are required for ticket resolution:

- Time to complete.
- Types of tasks as indicated by the custom filed in the ISDA::Admin queue. This metadata can be reviewed and changed by Pipeline.
- Every system affected by the change. This includes filesystem changes where software libraries are stored, runbooks updates, DNS and other network settings, and any other dependent system where we altered data or configuration

Release Strategy

DSPS' systems release alternately to NIST, Database Servers, or to Server Operations hosting for our own production support. System configurations are highly variable and therefore included in the runbook for each given system.

[Improvements to this process are under review. ed.]

Release Schedule

Productive Systems: Release is variable. ISDA project coordinators negotiate a release window by coordinating with Pipeline, the project team, and the operational team responsible for the given system. This holds for systems defined as either "staging" or "production."

Non-Productive Systems: Systems are installed in such a way that implementation teams should have full access to system configurations they need to modify. This is not accomplished by full root or full admin access. If permissions are not sufficient, coordinate change requests through Pipeline.

System Access Rules

- Only System Integrators in DSPS have shell access to "staging" or "production"

systems.

- System integrators perform all configurations and deployments to any staging or production environment that cannot be done from a web-management console.
- All developers can have shell access to prototype and development machines as users other than root.
- All developers can have access to any machine they support as the “logs” user, which will allow them read-only access to certain parts of the system.

Configuration Management

MIT does not have an automated configuration-management system so the term *configuration management* is used to denote process and controls, not technology.

[Improvements to this process are under review via a project to build a configuration-management system in collaboration with OIS. ed.]

Philosophy

All systems are prepared as if they will be rolled out to data-center operations for production support. The reality is that DSPS supports some of its own production systems but formal rollout to the support team is the ideal end state. Discipline in release management is guided by this principle.

Definitive Software Library

System integrators store definitive versions of system installations and standard components on file-system trogdor in ISDA's co-located environment. Access is tightly controlled and by command-line. File management is manual. Runbook documentation for any give system is stored in the ISDA wiki.

[Improvements to this process are under review via a project to build a configuration-management system in collaboration with OIS. ed.]

Application-Server Installation Standards

DSPS is moving away from the concept of standardized application "stacks," or in other terminology, standardized components and template layering. Ultimately we are counting on configuration management to handle system individuality.

In the interim, these are the versions of web-application software we are standardized on from the point of view of new installations and for which we have general runbooks that are not implementation specific.

General	Java Application Servers	PHP Application Servers
RHEL5.x Apache 2.x MySQL 5.x OpenSSL Development Environments Only: sFTP	JDK 1.6 SASHServer (SourceLabs Tomcat) Tomcat 5.5.x	PHP 5.2.x

Installation Constraints

Where we use standard packages (RPM, DEB), installed software stays in its default location. For custom builds, /usr/local is our preferred location for software installation.

Hand Off, System Installation

Development teams do not have shell access to accounts on staging or production environments. Only system integrators in DSPS and system operators in OIS have that access.

This constraint is by design, to force a hand-off of installation and configuration, in order to evaluate installation procedures as operationally sound prior to release to data-center operators. We call this "vetting the runbook."

Web Console Access

Some applications have web-consoles for use in provisioning and system configuration. For products that provide this feature, DSPS will grant administration privileges to staff on the development team based on the requirements of the responsible team's manager, coordinated through Pipeline.

System-Level User Accounts

We create standard system-level users on every machine: db, logs, repos, and www.

- Server applications run as these users, and not as root, for security reasons.
- Developers can use these accounts to access machines, since they cannot have root access. This access is permanent for development and test machines. It is temporary, for purposes of problem resolution, on staging and production machines.

The purpose of each account is as follows:

- "db" for MySQL or other database installations
- "logs" is to give developers read access to logs on staging or production servers.
- "repos" for Alfresco and other CMS installation
- "www" for Apache, Tomcat, and all other web-application server components

Only a fixed list of system integrators and system operators ever have root access to a web server. This list is fixed for systems where ISDA provides production support, variable for systems released to OIS.

Installation-Parameter Migration

For a long time, the data center required system integrators to install all custom software packages (those that were not part of the core OS package library) into the home directories for defined user types. This constraint has changed.

For any system going through update or revision, Pipeline will plan whether or not DSPS will take the extra time to put the system into a more industry-standard state for installation

location and packaging.

Source Control Management, Subversion

We assume the use of Subversion as provided by IS&T via the enterprise installation at svn.mit.edu. This rules will update as the capabilities of that system change.

Access Control

Our Subversion installation is integrated with Moira. Access control lists are stored there. Subversion's ability to send email notifications is also stored in Moira.

1. Always use a separate Moira lists for repository access control versus email notification.
   a) Do not define the ACL as usable as an email list.
   b) Do not define the email list as usable for access control.
   c) The ACL can be the same as one used to control access to AFS lockers, remember that this list's membership should not spread past the project team or people to whom they report.
   d) This email list is separate and discrete from human-to-human mail lists for projects. Not everyone on a project needs to be alerted about source-control updates. This is optional.

Directory Structure

Conceptual

[product]/[service]/[versioning]/[component]

Pseudo

[svn repos]/[top-level]/"trunk," "tags," or "branches"/[IDE Project or deployable unit]

Defined

Product: The arch, meta-project, brand, or other executive concept. This should not be a team or organizational unit.

Service: That which is useful, runs, or operates as a whole, a conceptual piece of an overall product that you are likely to manage as a coherent project.

Versioning: The standard SVN directories of "trunk," "tags," or "branches" denoting the primary version-control function of SVN.

Component: A collection of code that compiles as a single deployable unit (a war file, tarball), that which functions well as an IDE project, or both.

Example

We'll use a fictionalized multicomponent web service as an example.

- MitWebServices
  - GroupManager
    - trunk
      - GroupManagerWS
      - GroupManagerService
      - GroupManagerJNI
      - GroupManagerJMX
    - tags
      - Release- 0.1.1-20101104
      - Release- 0.2.0-20101104
    - branches
      - JohnDoesPrototype
      - PROD-0.1.1-20101104
  - CommonWsLibs
  - AnotherService
- SomeoneElsesRepos

By this strategy, you end up with a GroupManager source tree where the web service, underlying libraries, and instrumentation code are all managed and versioned together even thought they have different deployment paths.

You alse end up with a top-level repository that is used to manage the whole "product," which can be vague but a meaningful categorization when projects change hands.

The content of the GroupManager sub-tree contains only that which is local to operating that particular service, but shared or dependent within that service. CommonWSLibs are tools shared and common to the overall MITWebServices product, like AnotherService.

Tagging for Release

Under "tags," Version Capture

"Release"-[version number]-yyyymmdd

We do not specify any particular scheme for version numbers. Products, projects, and technology are too varied.

When you are done unit testing and prepared to move a service to a test system, tag that service directory as a "Release," including your local version number and date.

Under "branches," Production Support

Once the *tagged* release is approved for production deployment, *branch* it to a production release in the form:

"PROD"-[version number]-yyyymmdd

Again, discrete version numbers are up to the project team. Leave them out of the name if you are not using them or tracking by date.

Other Directories

This process is not designed to stifle the flexibility of source control. It is designed to give project managers and system operators a self-documenting way to find things across many projects. Beyond this basic organizational principle, other directories within a service, especially branching to prototype changes or tagging to capture mid-steps in the development process, is wide open.