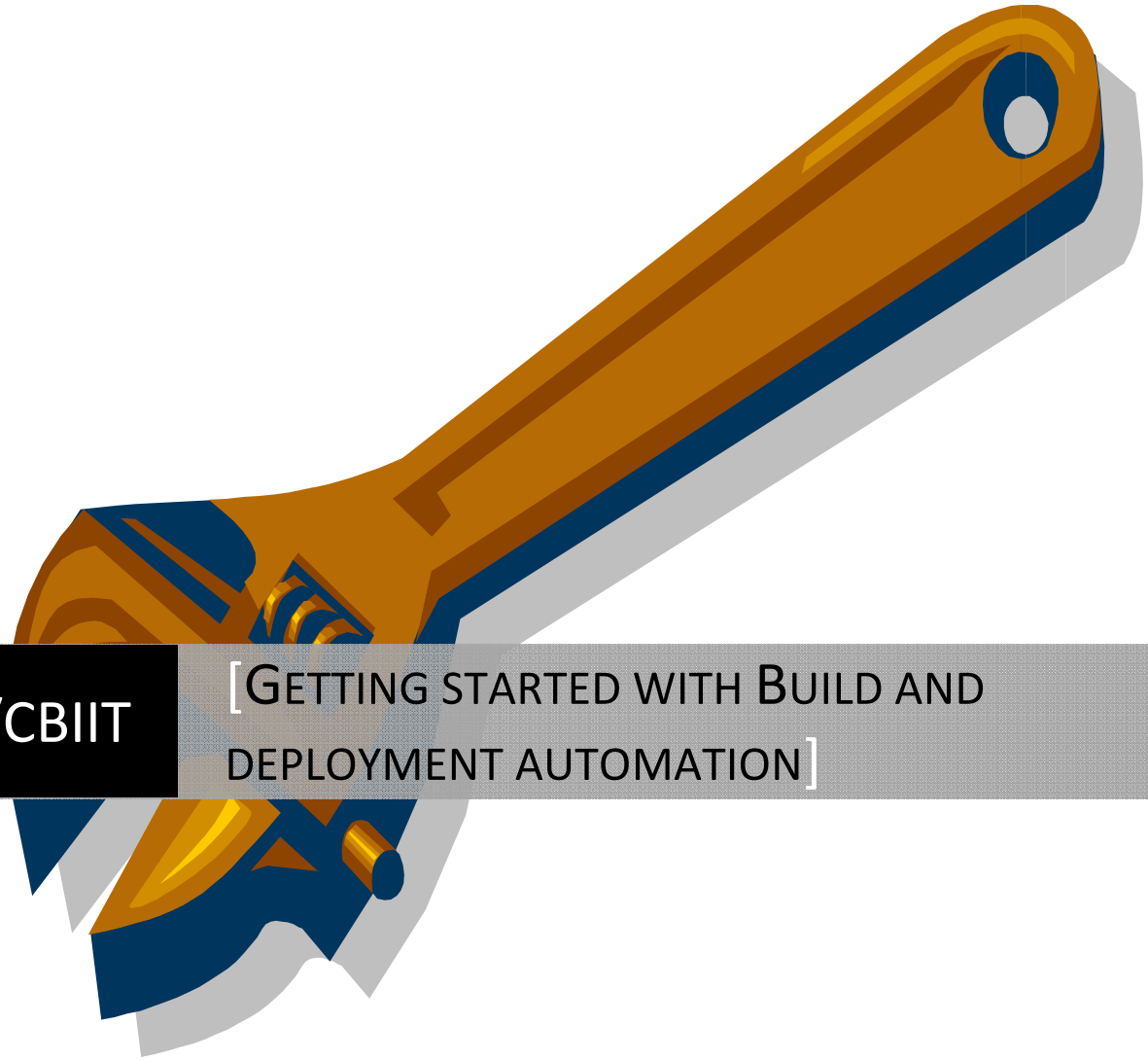
April 2008



NCI/CBIIT

[GETTING STARTED WITH BUILD AND DEPLOYMENT AUTOMATION]

National Cancer Institute | Build and Deployment Automation Team

Checklist

Item	Description
Are you/will you be using Subversion?	It's highly recommended teams use Subversion
List of property values	For DEV and QA. Work with the Systems team in determining DEV hostname, database hostname, authentication hostname.
Meet w/ Systems team	Ask systems team to setup virtual machine for CI environment. Request setup for DEV, QA, and STAGE environments.
Create text-based Systems request documents	See NCIA and caArray for an example. https://gforge.nci.nih.gov/svnroot/scm-private/trunk/caarray2/deployment/*.txt for an example.
Create tier-specific .properties files	All property names are the same. Values are different. See https://gforge.nci.nih.gov/svnroot/scm-private/trunk/caarray2/properties/*.properties for an example

Key Practices

Practice	Description
Use Continuous Integration	Build software with every change applied to version control repository
Commit code often (each developer at least once a day)	Task-level commits are preferred
Prevent Broken Builds	Run private builds prior to checking in code
Fix Broken Builds Immediately	Developer(s) responsible for breakage should be responsible for fixing it
Capable of delivering to DEV, QA, STAGE every day	It's very likely you won't be delivering to QA or STAGE everyday, but you should be capable of delivering every day. It is recommended you deliver to DEV everyday.
Capable of scorching DEV, QA, STAGE environment and rebuilding in automated fashion	Anyone on the team should be capable of standing up a target environment in less than one hour
Separate all tier-specific values into tier-specific .properties files	These files should be committed to a protected directory in SVN project repository
Same build run from AntHill Pro should be run the command line and vice-versa	Nothing in the build should be tier-specific.
No tier-specific build functionality	Although different targets may run in different environments, the build should not have **any** functionality that is not capable of

	running in **any** target environment.
Use a separate machine to run CI	Systems team will provision a virtual machine for this purpose
DEV, QA, STAGE environments should be similar	There should be no marked configuration differences between DEV, QA and STAGE environments. Full automation can assist in verifying this is the case
Use template property checker	See https://gforge.nci.nih.gov/svnroot/scm-private/trunk/ant/custom/ncicb
Promote binaries through tiers	Build in DEV, promote EAR/WAR from DEV to QA, QA to STAGE, etc.
Checkin all files necessary to run a complete build into Subversion repository	See Repository pattern at http://www.scmpatterns.com/book/pattern-summary.html
All values that vary between target environments must be tokenized in source/configuration files	There must not be in hard-coded values that vary between environments

Build Functionality

A build is much more than compilation and packaging. This table describes the functionality we recommend adding to your build scripts.

Function	**Description**
Compilation	Compile and package the Java source code
Dependency Management	Use a common repository to manage JAR and tool dependencies. An Ivy repository has been established at CBIIT
Database Integration	Execute DDL and DML as part of build process. Provide capability to rebuild database and test data automatically.
Database Migration/Incremental Database Modifications	Changing attributes/data for an existing database
Automated Tests	Unit, Component, System, Functional, Load & Performance, Security
Automated Inspections	Coding standards, Dependency Analysis, Cyclomatic Complexity, Duplication, Code Coverage
Deployment	Deploy/Configure to web container, grid service, etc.
Installation	Similar to Deployment, but creating installers for installation in user's environment
Documentation	JavaDocs, Doxygen (UML models, etc.), ER diagrams, build diagrams
Tool installation and configuration	Download, install and configure tools

	such as JBoss, MySQL and Globus in each target environment

Tools

Here are some of the tools your team can use to implement the practices identified above.

Tool	**Description**
Maven, Ant	Building the software
AntUnit	Write unit tests for Ant code using Ant
Ivy	Dependency management of JARs and other files
BDA macros • Database Integration/Migration • Local/Remote Deployment • Pre/Post-installation checks	Common framework of build, installation and deployment scripts for CBIIT
JUnit	Write automated unit and component tests in Java using JUnit
Eclipse	IDE. BDA has a common project file that can be utilized
DbUnit	Framework for writing component tests – specifically for seeding test data
Selenium	Framework for running automated web-based cross-browser functional tests
JMeter	Load testing tool
Fitnesse	Acceptance-testing tool
CheckStyle/PMD	Coding standards
Simian	Code duplication checker
JavaNCSS/Source Monitor	Check for cyclomatic complexity
JDepend	Tool for dependency analysis
Cobertura	Open source code coverage tool
JSch (for deployments)	Java Secure Channel is used for SCP and SSH commands
AntHill Pro	CBIIT build management team for promotion between target environments (DEV, QA, STAGE and PROD)
Hudson	Open-source Continuous Integration server used in CI environment

Procedures

An initial list of recommend procedures for your project.

Procedure	**Description**

Checkin procedures	Communicate the codeline policy for developers when checking in code
Build Promotion procedures	The steps for promoting from one environment to the next

Resources

NCI Common Library (Ivy repository of JARs and tools)	https://gforge.nci.nih.gov/projects/commonlibrary/
Build and Deployment handbook (see Docs|Policy Documents|Build and Deployment Handbook	https://gforge.nci.nih.gov/projects/scmapilot/
caArray2 SVN repo (BDA-enabled project)	https://gforge.nci.nih.gov/svnroot/caarray2
NCIA SVN repo (BDA-enabled project)	https://gforge.nci.nih.gov/svnroot/ncia
Template property checker (Check with BDA team on usage)	https://gforge.nci.nih.gov/svnroot/scm-private/trunk/ant/custom/ncicb

Target Environments

Target Environment	**Description**
Developer Workstation(s)	Each developer will manage his developer workstation environment. Each developer should be capable of running a full integration build on his machine with only an SVN client and JDK. Builds will occur very frequently in this environment.
Continuous Integration environment	Manage by each development team. Provisioned by the Systems team. A virtual machine will be established for each team and provide unfettered access. Builds will occur with every change to the SVN project repository.
DEV	Managed by Systems team, but development teams have certain level of access. It's recommend development teams run a daily build to this environment from AntHill Pro
QA	Managed by Systems team, but QA team will have a certain level of access. It's recommend QA run an on-demand build to this environment from AntHill Pro for each iteration.
STAGE	Managed by Systems team. Developers will not have nay access to this environment. However, the same build run in the other environments will run in this environment, but with different property values (known only to the Systems team)
PROD	Managed by Systems team. Developers will not have nay

	access to this environment. However, the same build run in the other environments will run in this environment, but with different property values (known only to the Systems team)

Examples

To use the BDA common macros, perform the following:

1. Checkout from https://gforge.nci.nih.gov/svnroot/automation/trunk/bda/ivy/
2. Add this Ant script to the beginning of your project's build script. It should be before ant taskdefs and after property definitions

```
<mkdir dir="lib"/>
<ant inheritAll="false" inheritRefs="false" antfile="bda-ivy-build.xml"
target="retrieve-bda">
  <property name="bda-utils.dir" value="${bda-utils.dir}"/>
  <property name="lib.dir" value="${lib.dir}"/>
</ant>
<import file="${bda-utils.dir}/bda-build-utils-1.0.xml" />
```

After adding this script, you can use the macros defined in bda-ivy-build.xml in your build scripts.