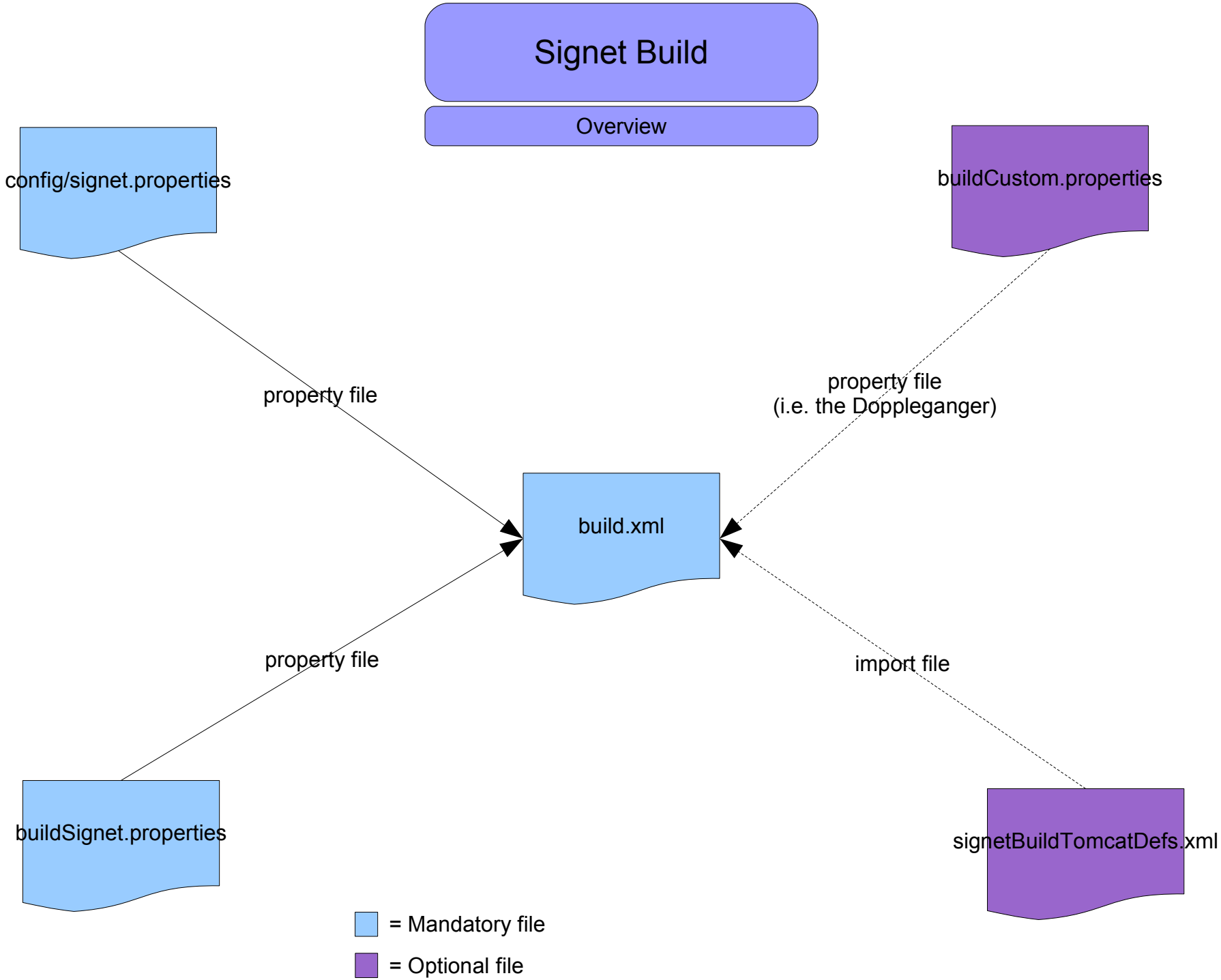
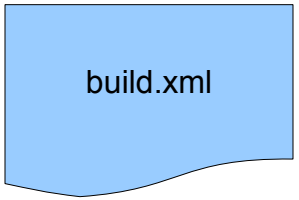
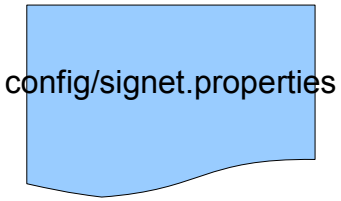

Signet Build
Overview
config/signet.properties
buildCustom.properties
property file
property file
(i.e. the Doppleganger)
build.xml
property file
import file
buildSignet.properties
signetBuildTomcatDefs.xml
= Mandatory file
= Optional file

Signet Build

Mandatory Files



The Ant build file. Defines numerous targets for generating Signet. This file imports several mandatory and optional property files that define source, destination, compile options, etc. Build.xml contains no user-modifiable variables. Customization should be done instead by changing variable values in the imported files.



The Signet Configuration file. This file is shared by both the Signet build and the Signet runtime. It contains properties required for both the build and runtime. The only properties a user may want to modify at build time are "signet_version" and "signet_basename". These would only be modified if the user wants to change their own version number and/or application name.

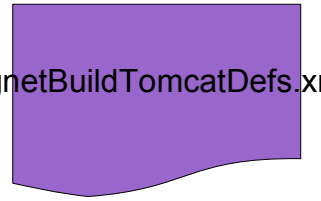
buildSignet.properties contains mandatory property values for the successful execution of the Signet build. The values of these properties are designed to work with the files retrieved from CVS, as is. Modifying any of these properties is not recommended, with the possible exception of the compiler options ("compile...")

Signet Build

Optional Files

The purpose of buildCustom.properties is to provide the ability to enhance, or add to, the standard Signet build. Developers interested in customizing Signet can use buildCustom.properties to include their changes without affecting, or being affected by, the Signet project. This is accomplished using the notion of a "doppleganger." Developers keep their files in directories that are external to the Signet project tree and add/modify a single line in build.xml to include their files into the Signet build.

If property overrides or dopplegangers are not required, then remove (or comment) the property reference to this file from build.xml.



signetBuildTomcatDefs.xml contains the properties and taskdefs required by Ant to interact directly with an installed and running instance of Tomcat. If Tomcat is not used, or if build-time deployment to Tomcat is not required, remove the import of this file from build.xml.

Signet Build

Prerequisites

CVS

A CVS client is needed in order to retrieve the Signet files from the repository. Several CVS clients are available for various platforms from the CVS Home page (http://ximbiot.com/cvs/wiki/index.php?title=CVS_Clients). Once the CVS client is installed and configured, the Signet project can be obtained with the following command line (GUI-based clients should use equivalent parameters):

```
cvs -z3 -d :pserver:anoncvs@anoncvs.internet2.edu:/home/cvs/i2mi co -r dist_v_1_0_3 signet
```

Note that the version may vary from "dist_v_1_0_3" as illustrated above. Also note that, at present, CVS does provide an easy way to query the repository for a list of available tags. Therefore, it will be necessary to rely on either the Signet mail lists or the Signet Wiki (https://wiki.internet2.edu/confluence/display/SignetWG/Home) for announcements of new Signet versions (i.e. CVS tags) if one uses the CVS command line. However, several of the Interactive Development Environments (IDEs), such as Eclipse (http://www.eclipse.org/) do provide this capability.

Java

In order to build Signet you will need Java JDK 5.0 (Java 1.5) or higher. It is not enough to have the Java runtime (JRE). You can quickly determine if you have the JDK, and which version, by invoking the Java compiler from the command line (a DOS prompt is illustrated below, but the same command should work on any operating system):

```
C:>javac -version
javac 1.5.0_06...
```

The Java JDK can be downloaded from Sun's download site: http://java.sun.com/javase/downloads/index.jsp

Tomcat

If Signet is to be installed into Tomcat, Tomcat must be installed and running in order for the Signet build to deploy Signet correctly. It is not necessary for Tomcat to be running on the same computer as the Signet build. However, the Tomcat server must be accessible by the build system. Tomcat can be obtained from Tomcat Home page (http://tomcat.apache.org).

(continues on next slide)

Signet Build

Prerequisites, Continued

Database

Although not required to build Signet, a database will be needed to run Signet. Currently, two databases are supported by the Signet DDL (SQL Data Definition Language) scripts supplied in the project's 'sql' directory: Hypersonic SQL and Postgres.

Note that the database referred to here is Signet's internal database used to keep track of permissions, assignments, et cetera. This database should not be confused with any database used to provide Subject information via the Subject API.

Hypersonic SQL is a free, in-memory, 100% Java implementation of an SQL database. It is probably not robust enough for a production deployment of Signet but may be advantageous for development work because of its size and speed. It can be obtained from their website: http://hsqldb.org/

Postgres too is a free SQL database but is much more robust and is suitable for production, client/server environments. It can be found at http://www.postgresql.org/

As an aside, Aquafold provides an excellent database development tool called Aqua Data Studio. It is a commercial product but they provide a free version (v4.7) as well. It runs on all of the popular operating systems. It may be found at their website: http://www.aquafold.com/downloads.html.

An alternative to Aqua Data Studio, if using the Eclipse IDE, is the Database Explorer View (not a Perspective!). From the Eclipse main menu select Window | Show View... | Other... Then expand the Data node and select Database Explorer. Then select OK. A new view window (or tab, depending on configuration) will open. Right-click the Connections node and select New Connection... Enter appropriate information for the database in use. For example, with a Postgres database running on the local machine, the following values can be used:

```
Select a database manager:    Generic JDBC | 1.0
JDBC Driver:                  Other
Database:                     signet_test2
JDBC driver class:            org.postgresql.Driver
Class location:               /Projects/signet/lib/postgresql-8.1-404.jdbc2.jar
Connection URL:               jdbc:postgresql://localhost:5432/signet_test2
User ID:                      xxxxxx
Password:                     yyyyyy
```

Select Finish. If properly configured, the Database Explorer should now display the database information.

Note that neither Aqua Data Studio nor Eclipse Database Explorer will work with the in-memory Hypersonic SQL database!

Signet Build

Configuring the Build

The Ant Build Script File build.xml

The Ant build script contains all of the public and private build targets used to build Signet. It contains no user-modifiable properties with the exception of the "include" directives (<property file=... and <import file=...) for the files mentioned below. Most users will find it sufficient to build either of two primary targets: "production" and "development". See slide "Running the Build" for more information.

Note that this is a monolithic build file -- it does not invoke any other Ant scripts. Any other build.xml files encountered are artifacts of previous Signet versions and may not work and are unsupported.

Signet Build- and Run-time Configuration File config/signet.properties

The properties defined in this file are shared by the Signet build and the Signet runtime. A developer may want to modify signet_version or signet_basename to create site-specific builds. Note that many of the build-time properties use these as part of their definitions.

Standard Build Properties File buildSignet.properties

All of the properties defined in this file are specific to the standard Signet build. The directory paths are all relative to the Ant build file which sits at the top of the Signet project file hierarchy. The only properties a developer may want to modify are those related the Java compiler (compile.*). If other properties are to be changed (cautiously!) it may be easier to instead override them using the Custom Build properties file, described below.

Signet Build

Configuring the Build, Continued

Custom Build Properties (the Doppleganger) File buildCustom.properties

The following properties are optional. Build.xml is configured to search for these properties and will use any that are defined.

`custom.java.dir`	Defines additional Java source files to include in the build
`custom.lib.dir`	Defines additional JAR files to be included in the build and the distribution
`custom.sql.dir`	Defines additional SQL files to be added to the Source distribution
`custom.config.dir`	Defines additional runtime properties and configuration files
`custom.webconfig.dir`	Defines additional runtime web configuration files
`custom.webimages.dir`	Defines additional runtime web image files
`custom.webmetainf.dir`	Defines additional manifest files to be included in generated JARs
`custom.webjsp.dir`	Defines additional Java Server Pages for deployment
`custom.webscripts.dir`	Defines additional JavaScript files
`custom.webstyles.dir`	Defines additional Style Sheets
`custom.webtiles.dir`	Defines additional tiles

All properties found in signetBuild.properties may be overridden by defining them in buildCustom.properties. However, this should be done with caution because it may cause the build to ignore required Signet files.

Configuring the Build for Tomcat Deployment File signetBuildTomcatDefs.xml

The following properties are used to deploy Signet to a running instance of Tomcat.

`tomcat.manager.username`	Defines the Tomcat manger user id
`tomcat.manager.password`	Defines the Tomcat manager user password
`tomcat.manager.url`	Defines the Tomcat manager URL
`tomcat.manager.signetpath`	Defines the Signet URL
`catalina_home`	Defines the Tomcat installation directory

Signet Build

Running the Build

Primary Build Targets

As previously mentioned, there are two primary build targets: production and development.

"production" performs a complete build up through the creation of a Tomcat-deployable WAR file (Web ARchive). It also creates the Javadocs for Signet's API (Application Programming Interface).
"development" is essentially a "production" build (without the Javadocs step) and additionally deploys the WAR file to a running Tomcat instance.

The output of the build is placed in a new directory at the Signet project root, called "dist-<basename>-<version>", where <basename> is replaced by the value of property "signet_basename" and <version> is taken from the property "signet_version" both defined in file config/signet.properties. The destination of the build output can be configured by overriding the property "dist_dir".
Two subdirectories are created under dist_dir: util and webapp. Util contains all the files (jars, config, properties, etc.) to run the Signet Utilities. It also contains the Javadocs, if they were generated. Webapp merely contains signet/signet.war.

Other Build Targets

The following command, entered from the Signet project root directory, will display a list of the available public build targets:

```
C:>ant -projecthelp
```

Note: Ant considers a target to be public if it has a "description" property defined. Public targets may be built directly from the command line. Whereas private targets may only be invoked internally from other targets.

Running the Build

The default target is "production". Simply entering

```
C:>ant
```

at the command prompt will build this target. Other public targets may be built by appending the target name to the invocation of ant. For instance, to build the "compile" target, enter the following at the command prompt:

```
C:>ant compile
```

Signet Build

Initializing the Signet Database

Overview

Signet requires some basic data in order to run. This includes information about Organization hierarchies, Permissions, Subsystems and Functions, and at least one Signet Proxy (i.e. super user). Several utilities and sample data are provided with the Signet project to populate the Signet database.

TreeXmlLoader

TreeXmlLoader provides a sample organization hierarchy. It must be run before SubsystemXmlLoader. For example, to run TreeXmlLoader from a DOS command prompt, enter:

```
C:>cd \Projects\signet\dist-signet-1.0.3\util\TreeXmlLoader
C:>run demo_tree.xml
```

A Unix shell script is also provided: `run.sh`

SubsystemXmlLoader

SubsystemXmlLoader provides sample Permissions, Subsystems and Functions, and their dependent data. It must be run after successfully running TreeXmlLoader. For example, to SubsystemXmlLoader from the Unix bash prompt, enter:

```
$ cd /Projects/signet/dist-signet-1.0.3/util/SubsystemXmlLoader
$ ./run.sh demo_biox_subsystem.xml
$ ./run.sh demo_library_subsystem.xml
```

These commands will load two subsystems into the Signet database.

Signet Build

Initializing the Signet Database, Cont.

SubjectFileLoader

At present, for the purpose of convenience, the Signet project utilizes the Signet database as a provider of Subjects through the use of several tables and a JDBCSubjectAdapter in the SubjectAPI. These tables are not part of the official Signet database but demonstrate how one might implement their own JDBC subject provider. SubjectFileLoader must be successfully run before SignetProxy, below. For example, to run SubjectFileLoader from the DOS command prompt, enter:

```
C:>cd \Projects\signet\dist-signet-1.0.3\util\SubjectFileLoader
C:>run demo_subject.txt
```

This will create numerous example Subjects.

SignetProxy

Signet requires that a real Subject be provided as a Proxy to act as the Signet super-user. The Subject can be any Subject accessible through the Subject API(s) available at the time SignetProxy is run. The sample Subject data used above in SubjectFileLoader contains one Subject, "kmart", that has been defined for this purpose. For example, to designate kmart as the Signet Proxy using the Unix commands, enter:

```
$ cd /Projects/signet/dist-signet-1.0.3/util/SignetProxy
$ ./run.sh grant kmart
```

This will grant kmart permission to designate Subsystem owners. They, in turn, can grant permissions to other Subjects.