sprout ant more
sprout more ant
ant more sprout
more sprout ant
ant sprout more
more ant sprout

sprout

Ant documentation for SProUT

{ulrich.schaefer,daniel.beck}@dfki.de

2005-11-22

0. Introduction

Apache ant is a platform-independent build tool for Java-based applications. It is entirely written in Java and uses XML syntax to describe the build process and its dependencies. Ant has been developed by the Apache project and is available freely from http://ant.apache.org

Ant has been chosen for SProUT system builts because it does not only provide mechanisms for compiling and running Java code, but can also be used to manage, compile and distribute the linguistic resources, support remote repositories, compile JavaCC grammars, package jar and zip archives, create and compare configuration files, build documentation etc.

The central ant build file for SProUT (src/build.xml in the CVS) is described in this documentation, but – as ant build files can build on other ant build files (got that?) – build.xml can also form the basis for your own project definitions.

Ant 1.5.4 is part of the SProUT CVS sources and hence need not to be installed separately. The central SProUT build file src/build.xml has been developed under Linux. A few targets currently only run on Linux (flop-based targets and targets using sed like sprout2hog), the rest should be platform-independent.

You are encouraged to contribute further targets, extend the documentation etc.!

Please note that some targets and properties may be renamed for consistency in the near future. We apologize for inconsistencies and non-orthogonality of the existing build targets, which is in great parts caused by the very different modules and compilers that this ant build file tries to unify and make uniform. Also, some rarely used targets or resources may not work/have not been tested for a long time. In

addition, not all resources are available for every language, hence, some targets may not work because of missing resources. Only English and German grammars have been tested well.

Please report ant-related bugs or send suggestions or fixes to the authors.

1. Ant for SProUT – Getting Started

a) Checking out the sources of SProUT

Create a new directory for your local CVS directory, e.g.,
```
>>  mkdir sproutcvs
```

Set the location of the CVS repository
```
>> export CVSROOT=/project/cl/sprout/cvsroot
```

To checkout files or directories from CVS, go to your local cvs repository:
```
>> cd sproutcvs
>> cvs checkout src
```

Later on, to get updates from CVS, use
```
>> cvs update -d src (or on more specific subdirectories or files)
```

When you have checked out the SProUT sources, do
```
>>  cd src
```

b) Adjust the ant start script

First, edit the script file src/ant (src/ant.bat for Windows)

You may want to adjust the JAVA_HOME and ANT_HOME paths. JAVA_HOME must point to you Java installation, e.g., JAVA_HOME=/lt/pkg/j2sdk.1.4.2_07, otherwise ant will complain even if javac and java are in your search PATH. If JAVA_HOME is already properly set in your system environment, leave it commented in the ant start script.

You may also adjust your ANT_HOME to an existing ant installation (ant also belongs to the SPRoUT sources, that's why leaving that as ANT_HOME=./apache-ant-1.5.4 will suffice in most cases).

c) Start the SProUT IDE

To start ant with the SProUT standard target, run
```
>> ./ant
```

Then the IDE for developing grammars should start (and be compiled if not yet done).

The default build file is build.xml in the current working directory which does not need to be specified on command line. Other build files, e.g. your own project-specific ones, may be chosen using the -f ‹buildfile› command line option.

2. Running targets from the ant buildfile build.xml

To start ant target, run
>> ./ant ‹target(s)›

The following command lists available SProUT targets in the central build file
>> ./ant -projecthelp

Some targets requires parameters. You can pass them to ant by using -D‹param›=‹value›, e.g.
>> ./ant -Dlang=de compile_ne

For a short overview of ant command line syntax see
>> ./ant -help

For a complete overview of all targets in the the ant build file, run
>> ./ant antdoc

and open the generated HTML documentation in
“run/doc/antdoc/buildxml/index.html” with your favorite HTML browser

3. Directory structure

The build.xml ant script creates a run/ directory structure in parallel to the CVS mirror in src/ reflecting the CVS directory structure where appropriate. The grammar resources in src/grammar will by default be compiled to run/data, but their subdirectories bear the same names, e.g. the compiled counterparts of src/grammar/tdl/en/en_types.tdl can be found in run/data/tdl/en/en_types.grm after calling ./ant compile_tdl -Dlang=en.

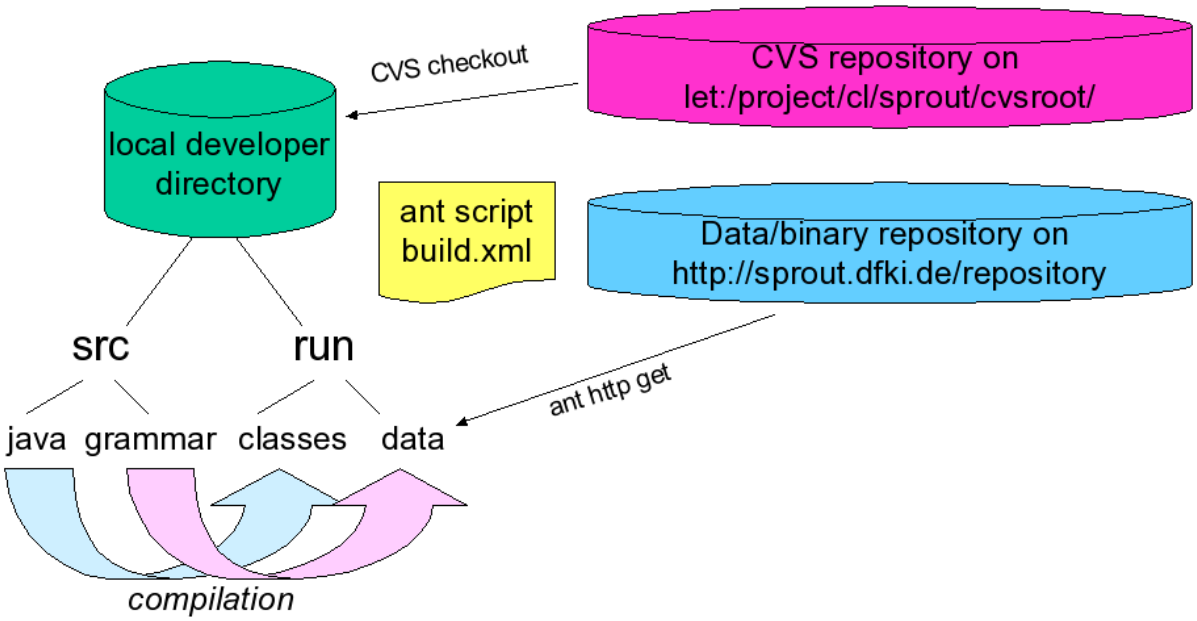


Figure 1 SProUT build architecture

In the following tables, ${basedir} denotes the directory where the user checked out the CVS sources, i.e., the parent directory of src.

The default properties (as defined in CVS) are in the file src/build.properties.

User-specific and superseding properties can be defined in user.properties (one directory level higher, i.e., not in CVS).

Directory properties for source files

Property	**Default value**	**Description**
src.dir	${basedir}/src	Base for all sources (CVS root)
srcdata.dir	${src.dir}/grammar	Base for grammar sources etc.
srcjava.dir	${src.dir}/java	Base for java sources
srclib.dir	${src.dir}/lib	Base for libraries needed for compilation

Directory properties for generated files

Property	**Default value**	**Description**
run.dir	${basedir}/run	Base for all generated (compiled) files
rundata.dir	${run.dir}/data	Base for compiled grammars & resources
doc.dir	${run.dir}/doc	Base for generated documentation
classes.dir	${run.dir}/classes	Directory for generated Java class files
apidoc.dir	${doc.dir}/api	Base for generated Javadoc

Property	Default value	Description
runlib.dir	${run.dir}/lib	Directory for generated/copied runtime libraries (non-Java)
runtmp.dir	${run.dir}/tmp	Temporary directory for compilation and runtime
runjar.dir	${runlib.dir}/java	Directory for generated/copied runtime libraries (Java)
runbin.dir	${run.dir}/bin	Directory for copied runtime binaries (non-Java), e.g. flop

4. Overview of the most frequently used targets

This table shows the most frequently used targets defined in build.xml. There are additional, mostly auxiliary targets. For details, see the generated antdoc HTML documentation.

Target name	***Parameters (-Dparam=value)***	***Description***
Core targets		
ide		compile and run grammar IDE
compile		compile grammar IDE
jar		compile and package sprout.jar
runtimejar		compile and package sprout-runtime.jar
runtimejar_with_doc		compile and package sprout-runtime+doc
clean		delete compiled/generated files in run/
regcompilergui		GUI for regular compiler
installergui		GUI for runtime package installation
sprout2hog	out.dir	build SProUT runtime component for HoG
fs2latexjar		build fs2latex.jar package
fsapplet		generate FS renderer applet jar
runtimetest	cfg.file, input.file	test SProUT runtime
runapplet	xml.file	displays SProUTput XML in browser
Grammar compilation		
compile_ne	lang, subgrammar, charset, project, projectname	compile xtdl, tdl, tok, ext. gazetter for ne

Target name	***Parameters (-Dparam=value)***	***Description***
compile_grammar	lang, project, projectname, projectfile, subgrammar	compile xtdl (.sgr) grammar to .fsm
compile_tdl	lang, project, tdlname, subdir	compile tdl type hierarchy to .grm
compile_tokenclasses	lang, ... (see doc)	compile a single tokenizer class to .fsm
compile_extended_gazetteer	lang, charset, project, insensitive	compile the extended gazetteer files for a language a to .sgz file. If insensitive is set to yes, the gazetteer will be compiled case insensitive – otherwise it will be compiled case sensitive
generate_extended_gazetteer	lang, charset, insensitive	generate temporary file to compile the extended gazetteer. If insensitive is set to yes, the gazetteer will be compiled case insensitive – otherwise it will be compiled case sensitive
clean_data		delete compiled tdl, xtdl, tokenizer files
clean_all_data		delete compiled tdl, xtdl, tokenizer, lexicon, (extended)gazetteer files
Copying/downloading resources		
get_file	repository, dest.file, repository.file	generic target to copy/download file from SProUT binary file repository
get_lexicon	lang, repository	copy/download compressed lexicon file
get_all_lexicons	repository	copy/download all compressed lexicon files
get_flop	repository	copy/download flop executable
Generating documentation		
javadoc	apidoc.dir	generate Javadoc of all classes in the CVS
javadoc_runtime	apidoc.dir	generate Javadoc of the runtime system only
antdoc	antfile, destdir	generate antdoc of the build.xml file
get_apachedoc	dest.dir	fetch ant documentation for ant 1.5.4
ant2dot	format, out.file	generate visual dependency graph of ant targets in build.xml
Experts only		

Target name	*Parameters (-Dparam=value)*	*Description*
xtdl_javacc	javacc.dir	Generate XTDL parser with JavaCC
autotest_all	out.dir	automatic system build and test (under cons.) for english and german
autotest	lang, config.file	automatic system build and test for language ${lang}
mmorphlex_cvs	lang	generated compressed lexicon file
sproutproject_to_install	out.dir, projectfile, subgrammar, lang	generate an install package from .spj file (s)
install_sprout_package	in.dir, out.dir	install a generated install package
jtaco_batch		run JTaCo in batch mode with a test text

5. Adding a new target to the central build file

You are hereby encouraged to do so, as long as the new target is relevant and useful also for other SProUT users or developers. Alternatively, you could create your private build file that calls dedicated targets in build.xml.

To see how to write targets, look at section “References/Official documentation” at the end of this document.

IMPORTANT: Antdoc (the target that can be used to generate a documentation of build.xml à la Javadoc) needs targets written in this form:

```
<!--#######target_name###################################-->
<target name="target_name" description="desc. f. -projecthelp">
<!-- Description : Write here what the target does -->
<!--Parameter: ${Prop1} Description of Prop1-->
<!--Parameter: ${Prop2} Description of Prop2-->
 ...
<!--Parameter: ${Propn} Description's of Propn-->
...
</target>
```

Insert targets only under that line :
`<property file="${basedir}/src/build.properties"/>`
else very bad things can happen, because some variables of your target might not been initialized .

Ant has got some limitations : Once an ant property has been set, it is immutable - it can neither be changed nor cleared. However some of the types of ant "recursions" (Ant invoking ant) allow for the wholesale inclusion or exclusion of properties to the called Ant task. If you tell it to drop all properties, you then have to explicitly pass any properties you didn't want dropped.
This is why if you write an target that will be called more as once from an other target, you should put it in the build file "sub_build.xml", and call it from build.xml with :

```
<ant antfile="name_of_the_target">
   <property name="property1" value="value_of_property1"/>
   <property name="property2" value="value_of_property2"/>
   ...
   <property name="propertyn" value="value_of_propertyn"/>
   ...
</ant>
```

Please follow these patterns – so everyone will be able to understand and see how to use your target.

6. Creating your own build files

You may also create your own ant project definition files which may refer to the central CVS-based build.xml using the <ant> task. You can overwrite selected properties of the central build file, cf. section **build.properties and other Properties** below.

7. Adding a new XTDL grammar to the CVS

The CVS directory for XTDL grammar has this structure :

src/grammar/xtdl/${subgrammar}/${lang}/

- lang is the two-letter ISO language code (cf. Appendix 1 – ISO 639 language code)
- subgrammar is the name of the language you wanted to add (e.g., ne for named entity grammars)

The name of the main file of the default project should be : ${lang}.spj
The name of the main file of an other project should be : ${lang}_${project}.spj

If you follow this rule, you can use the targets defined in build.xml – so you can avoid writing extra targets.

Example:
If the chunk grammar for Esperanto is located in
src/grammar/xtdl/chunk/eo/

and the main projectfile name is
src/grammar/xtdl/chunk/eo.spj

then you can simply compile the grammar with
./ant compile_grammar -Dlang=eo -Dsubgrammar=chunk

If the projectfile doesn't belong to the default project, but to the “intern” project, its name is
src/grammar/xtdl/chunk/eo_intern.spj

And you can compile the grammar then with
./ant compile_grammar -Dlang=eo -Dsubgrammar=chunk -Dproject=intern

(to be extended for TDL, extended gazetteer, tokenizer)

8. build.properties and other Properties

There are at least 3 locations where properties can be defined:
a) on the command line

b) in the build.properties file
c) in the build.xml file

The general rule is that a property specified on command line overwrites a property with the same name specified in the build.properties file which in turn overwrites a property with the same name in the build.xml file. This default behavior can be circumvented e.g. with special attributes in the antcall task, cf. the ant documentation for details.

Typically, settings like the binary file repository location which are set only once for a user environment (e.g. repository=local within DFKI NFS, or repository=net outside DFKI NFS), go to the build.properties file, while target-specific properties like lang are specified on command line. Most properties (parameters) in the build.xml targets have a default value (like en for lang) which can be overwritten from outside.

9. Generating Documentation

The javadoc and javadoc_runtime targets create javadoc in ${apidoc.dir}.

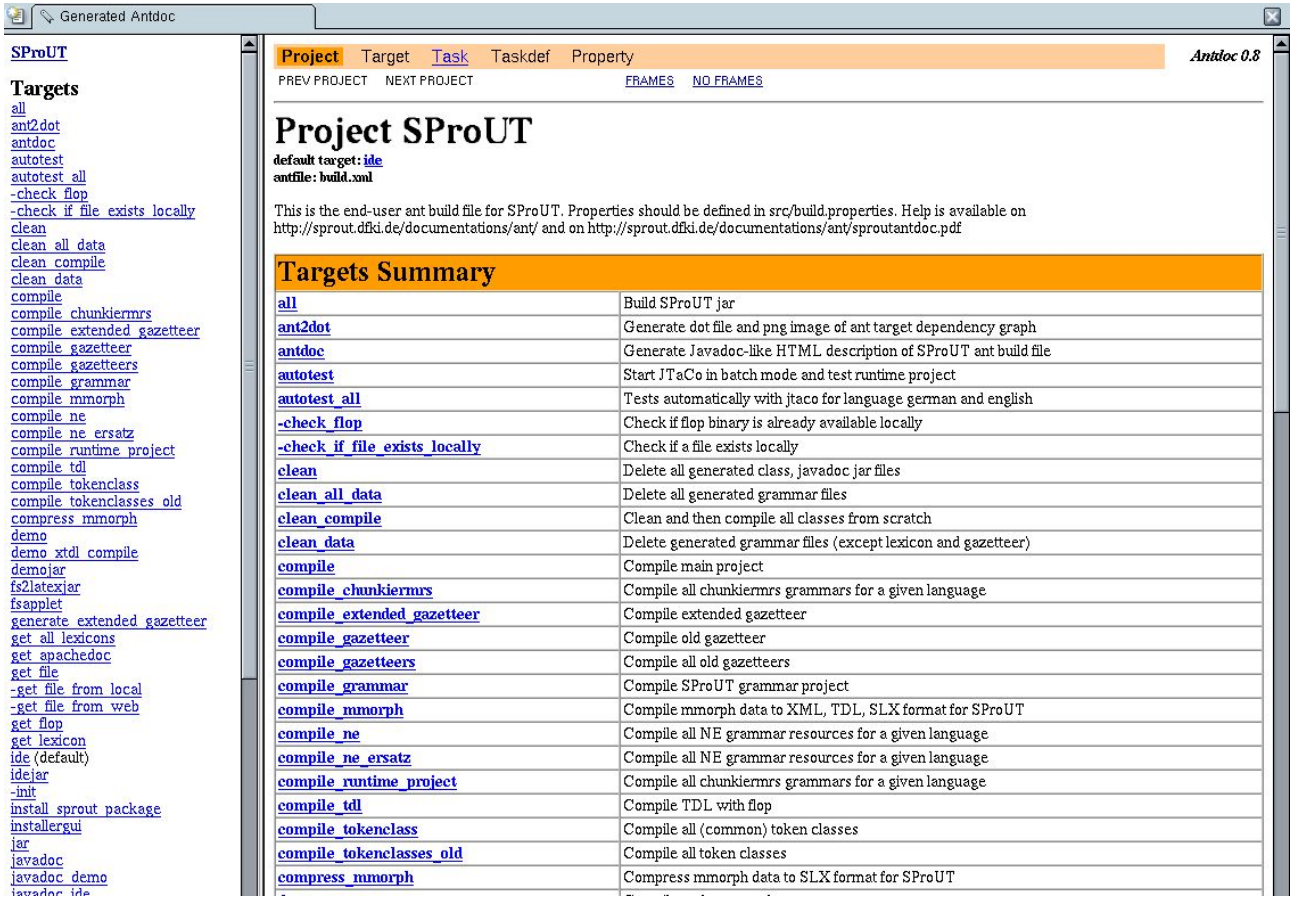


Figure 2 Documentation generated by antdoc

The antdoc target generates an HTML documentation of the build file build.xml in ${doc.dir}/antdoc/ similar to Javadoc, Figure 2 shows an example. The antdoc target may also be used for generating documentation of custom build files when specified using the -Dantfile= and -Ddestdir= parameters.

10. Integration of the SProUT build file with Java IDEs

- Eclipse 3 JDT has built-in ant support. Choose ant as builder instead of the built-in one. You will see available targets in a tree view, and properties may be defined from within Eclipse JDT build definitions.
- there are similar plugins for JBuilder and probably also for NetBeans

11. Redirection of Ant Logging

Ant can be configured to output its target and task logging to log4j.

./ant -listener org.apache.tools.ant.listener.Log4jListener -Dlog4j.
configuration=file:src/config/log4j.cfg <target>

12. References/Official documentation

Ant manual (for Ant version 1.5.4)
http://sprout.dfki.de/documentations/ant/manual/index.html

Ant task guidelines
http://sprout.dfki.de/documentations/ant/ant_task_guidelines.html

Ant task reference (pdf)
http://sprout.dfki.de/documentations/ant/appendix_e.pdf

Ant in Anger
http://sprout.dfki.de/documentations/ant/ant_in_anger.html

FAQ
http://sprout.dfki.de/documentations/ant/faq.html

Troubleshooting
http://sprout.dfki.de/documentations/ant/problems.html

Ant resources (external links)
http://sprout.dfki.de/documentations/ant/resources.html

Related projects
http://sprout.dfki.de/documentations/ant/projects.html

External tools and tasks
http://sprout.dfki.de/documentations/ant/external.html

If you need to download and install ant, the official site is
http://ant.apache.org

13. CVS documentation

E.g., online book „Open Source Development with CVS“ (3rd edition):
http://cvsbook.red-bean.com/

Appendix A: ISO 639 codes (only selected codes, mostly European and Asian)

These codes should be used for language identifiers in CVS, ant target names and lang property values.

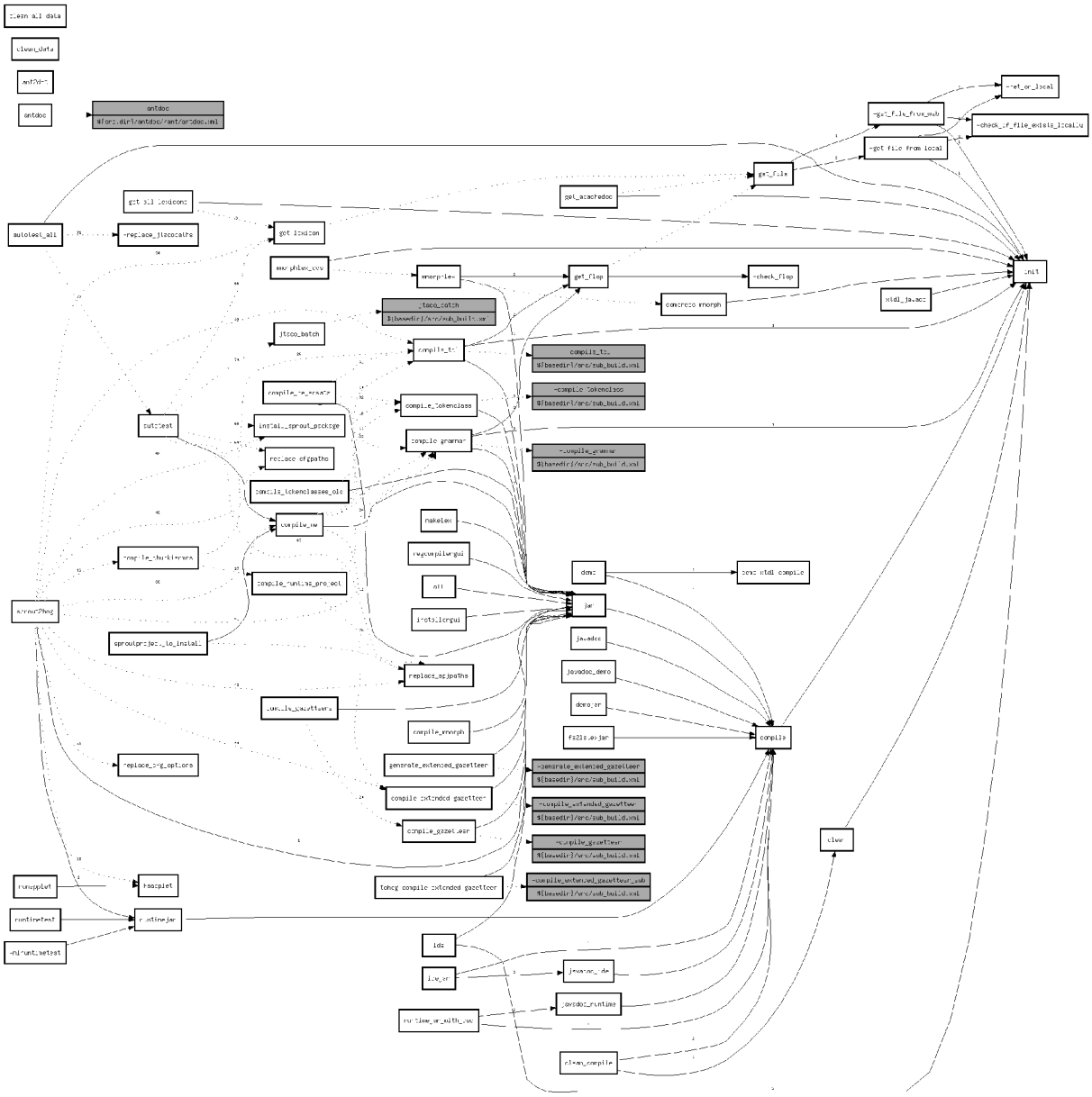
af	Afrikaans
ar	Arabic
be	Byelorussian
bg	Bulgarian
br	Breton
ca	Catalan
cs	Czech
cy	Welsh
da	Danish
de	German
el	Greek
en	English
eo	Esperanto
es	Spanish
et	Estonian
eu	Basque
fa	Persian
fi	Finnish
fr	French
fy	Frisian
ga	Irish

gd	ScotsGaelic
gl	Galician
ha	Hausa
he	Hebrew
hi	Hindi
hr	Croatian
hu	Hungarian
id	Indonesian
is	Icelandic
it	Italian
ja	Japanese
ka	Georgian
kl	Greenlandic
kn	Kannada
ko	Korean
ku	Kurdish
la	Latin
lt	Lithuanian
lv	Latvian
mk	Macedonian
mo	Moldavian
mt	Maltese

nl	Dutch
no	Norwegian
oc	Occitan
pl	Polish
pt	Portuguese
rm	Rhaeto-Romance
ro	Romanian
ru	Russian
sa	Sanskrit
sh	Serbo-Croatian
sk	Slovak
sl	Slovenian
sq	Albanian
sr	Serbian
sv	Swedish
tr	Turkish
uk	Ukrainian
vi	Vietnamese
yi	Yiddish
zh	Chinese

Technical contents of ISO 639:1988 (E/F)
"Code for the representation of names of languages".
The Registration Authority for ISO 639 is Infoterm, Osterreichisches Normungsinstitut (ON), Postfach 130, A-1021 Vienna, Austria.

Appendix B: Target Dependencies in build.xml (generated with ant2dot)



Appendix C: built-in properties

The first five properties are defined by ant, the rest are defined by the Java VM that runs ant (see http://java.sun.com/j2se/1.4/docs/api/java/lang/System.html#getProperties())

Property name	description
basedir	the absolute path of the project's basedir (as set with the basedir attribute of <project>)
ant.file	the absolute path of the buildfile
ant.version	the version of Ant
ant.project.name	the name of the project that is currently executing, it is set in the name attribute of <project>
ant.java.version	the JVM version Ant detected, currently it can hold the values "1.1", "1.2", "1.3" and "1.4"
java.version	Java Runtime Environment version
java.vendor	Java Runtime Environment vendor
java.vendor.url	Java vendor URL
java.home	Java installation directory
java.vm.specification.version	Java Virtual Machine specification version
java.vm.specification.vendor	Java Virtual Machine specification vendor
java.vm.specification.name	Java Virtual Machine specification name
java.vm.version	Java Virtual Machine implementation version
java.vm.vendor	Java Virtual Machine implementation vendor
java.vm.name	Java Virtual Machine implementation name
java.specification.version	Java Runtime Environment specification version
java.specification.vendor	Java Runtime Environment specification vendor
java.specification.name	Java Runtime Environment specification name
java.class.version	Java class format version number
java.class.path	Java class path
java.library.path	List of paths to search when loading libraries
java.io.tmpdir	Default temp file path
java.compiler	Name of JIT compiler to use
java.ext.dirs	Path of extension directory or directories
os.name	Operating system name
os.arch	Operating system architecture

Property name	description
`os.version`	Operating system version
`file.separator`	File separator ("/" on UNIX)
`path.separator`	Path separator (":" on UNIX)
`line.separator`	Line separator ("\n" on UNIX)
`user.name`	User's account name
`user.home`	User's home directory
`user.dir`	User's current working directory