FreeCol Documentation
Developer Guide for Version 0.13.0

The FreeCol Team

July 9, 2022

Contents

Chapter 1

How to become a FreeCol developer

1.1 The goal of our project

We are aiming towards making a clone of the old computer classic "Sid Meier's Colonization".

New features should in general not be added before we reach FreeCol 1.0.0, unless they are implemented as optional features using the class "GameOptions".

(Note that if you add a game option, add a default value setting in FreeColServer.fixGameOptions so that use of the option does not break old games. The same applies to client-options, in ClientOptions.fixClientOptions.)

The big exception to this rule is the our client-server model that will allow players from all over the world to compete in a game of FreeCol.

Read more about FreeCol on the FreeCol website.

1.2 SourceForge project site

You should visit and get familiar with our project site at SourceForge.

This site contains trackers for bugs and features requests, a task manager and lots of other important services.

We expect developers to use this site regularly when making changes to the codebase.

1.3 How to find tasks

While FreeCol has no working roadmap to the 1.0.0 release, you may find available tasks in the bug and feature request trackers — just grab any task you are planning to work on in the immediate future by posting a comment stating you intend to work on the specified task or by simply assigning the ticket to yourself.

Please create a new bug report/feature request for tasks that are not listed in a tracker (and before you start working on them). Please also remember to post a comment when you are done, or if you are unable to complete the work.

Major changes to the code should be discussed on the developer's mailing list before they are implemented. This ensures that your work will not be in vain if somebody else knows a better method of complete the objective or needs to offer feedback so as to not break some other portion of the codebase.

Please try to use the standard Sun Java coding style (link omitted, Oracle keeps moving it). We do not rigidly enforce it and tolerate minor variants, but it makes for a more browseable codebase if the style does not vary greatly.

1.4 How to use the trackers

If you are a full developer (i.e. with write privileges), this is how a bug tracker item should be updated while you are working:

1. Assign yourself to the tracker item before you start working on it.
2. Please verify that no duplicate entry has been posted.
    - If you can find a duplicate and the bug has *not* been fixed, please set its status to "Closed-Duplicate", its milestone to "Unspecified", and post a comment with the ID of the other tracker item (add a comment to the other tracker item as well).
    - If you can find a duplicate and the bug has been fixed, use the "Closed-out-of-date" status to close the tracker item.
3. Set the item's status to "Open-Needs-Info" if you require input from the person originally submitting the item.

4. Set the item's status to "Open-WWC1D" if the fix requires determining what Col1 did ("What Would Col1 Do?").

5. If you are unable to complete the item: assign it back to "None" and make a comment describing any problems relevant for another developer. The milestone of open bugs should be "Current".

6. If you successfully commit a fix for a bug, set the milestone to "Fixed-trunk" and the status to some form of "Closed" if you are certain of the fix and the bug was introduced during the current development cycle, or "Pending-Fixed" if the bug was present in a release. The intent is that at release time all "Fixed-Trunk" bugs can be re-labelled as "Fixed-*release*". Please also write a comment telling that the work is done, and it is helpful to refer to any commit/s where relevant changes occurred.

This is how a feature request item should be updated while you are working:*(Needs updating since the sourceforge migration)*

1. Assign yourself to the tracker item before you start working on it.

2. Please verify that no duplicate entry has been posted.

   - If you can find a duplicate and the feature has NOT been implemented: Please use the group "Duplicated", set the status to "Closed" and post a comment with the ID of the other tracker item (add a comment to the other tracker item as well).
   - If you can find a duplicate and the feature has been implemented: Use the group "Out of date" and close the tracker item.

3. Set the item's status to pending if you require input from the person originally submitting the item.

4. Set the group to "Accepted" if you decide that this feature request should be added before the next release (please discuss on the mailing list if there are any reasons not to include this feature).

5. If you are unable to complete the item (or you think someone else should be implementing it): Assign it to "None" and make a comment describing any issues relevant for another developer.

6. After you have completed the work: Set the group to "Added" and the status to "Closed". Please also write a comment telling that the work is done.

You can use any suitable canned response instead of writing a comment.

1.5 Mailing lists

Our primary means of communication is the developers mailing list: freecol-developers@lists.sourceforge.net

You can (and should) subscribe to this list.

1.6 Git

Git is the tool we are using to manage the changes within our source code tree. This system makes it possible for all developers to have their own full copy of the project. Git supports synchronization between the central version of the code ('the repository') and the local copies. Git also makes it possible to undo changes that were previously committed to the repository.

This page describes how you can start using Git and get a working copy of the code (without commit privileges).

You can use `git pull` for updating an existing working copy. Changes can only be applied by those who have write-access, so you may need to either send the changes to the developer mailing list or use the patch tracking system.

1.7 Compiling the code

FreeCol uses the *Apache Ant* build system for compiling the game. You can download a copy of Ant program from the Apache Software Foundation website.

After Ant has been installed, you simply type `ant` in the top directory of your FreeCol "working copy" in order to compile the game. The file `FreeCol.jar` will then be generated, and you can start the game simply by running the following command:

```
java -Xmx1G -jar FreeCol.jar
```

Once Ant builds the JAR file, you can rebuild FreeCol as needed any time a modification to the code requires testing. Using Ant to create a new version of the FreeCol file simply requires another command of `ant`. Only the files changed since the last compilation are rebuilt. This can lead to the occasional bug, so it is a good idea to run `ant clean` between all non-trivial builds. This starts the new build from scratch.

1.8 Using an IDE

Most FreeCol developers don't seem to use an IDE, so there is no "official" setup available. However, in the config folder, you can find contributed configuration files for both NetBeans and Eclipse. The following information has also been contributed by players that do use an IDE.

The FreeCol gitignore file (a list of files and directories that are ignored by Git) includes support for the following IDEs: Eclipse, JetBrains, JDeveloper, and NetBeans. Certain Linux and Windows specific files are also ignored.

1.8.1 Using Eclipse (thanks to "nobody")

This section is out of date since we migrated from svn to git. Leaving as-is for now in the hope the procedure is similar.

Since I'm quite a fan of the Eclipse IDE, I thought I would share my experience with building FreeCol in Eclipse on the Windows platform.

I assume that you have installed JDK, Eclipse and Git (both Eclipse plug-in and stand alone client). Make sure that your path environment variable contains both the JDK and SVN client directories.

First, add a new repository location in Eclipse in the SVN Repositories view, for the FreeCol repository. Leave all the other settings unchanged and click Finish.

Select either 'trunk' or the branch you want to build, right-click on it and select 'Find/Check out as...'

In the Check Out dialog, make sure the option 'Check out as a project configured using the New Project Wizard' is selected, and click Finish.

Select 'Java Project' in the 'Java' category, and click Next. Name your project (FreeCol is an obvious choice). Leave all the other options as is, and click Finish.

Eclipse starts to copy all the files from the repository. Depending on the server and your connection, this may take from 1-10 minutes. Get a cup of coffee or a glass of cold milk while you wait.

After downloading has finished, you should see your new project in the Project Explorer. Right-click on the project and select 'Configure Build Path...' from the 'Build Path' sub-menu.

First off, let's make sure that Eclipse has detected the source file folder 'src'. Select the 'Source' tab, and make sure there is an entry with the name [project name]/src. If not, add it. Don't close the window, as we need to make other changes here.

Next, we have to add the external jar files to the project, so Eclipse can properly verify the code. Select the 'Libraries' tab, and click the 'Add JARs...' button. Browse to the 'jars' subfolder in the FreeCol project, and select all the jar-files by holding down the CTRL-key. Click OK, and OK again.

Eclipse should now be able to properly build the project without any errors. If not, fix it. However, we don't actually want Eclipse to do this, since we instead want to use the Ant build file from the repository.

Right-click on the project, and select 'Properties...' all the way at the bottom of the menu. Select 'Builders' in the menu to the left. You should now see one entry in the list, named 'Java Builder'. This is the default, built-in java builder in Eclipse. Click 'New...' to create our Ant builder instead. Select 'Ant Builder' from the list and click OK.

In the configuration dialog, click the 'Browse Workspace...' button the 'Buildfile' section. Click on the FreeCol project, and select 'build.xml' from the list on the right. Click OK. Click OK again, and the Ant builder is created. You can keep both builders active at the same time, but if you want to save processing power, you can uncheck the 'Java Builder'. Eclipse will warn you about doing this, but don't be alarmed, you can always turn it on again.

Click OK. If you have activated Automatic building in Eclipse, Ant should start building the project right away. Possible errors could be, that Ant cannot access either the java compiler or a stand-alone svn client. In either of these cases, make sure you added the right directories to your path environment variable.

If the build went succesfull; congratulations. Open the project folder in the file system, and you will see 'FreeCol.jar' in the root folder. Since this is an executable jar file, you can double-click it and launch the game right

away. Enjoy.

1.8.2 Using NetBeans (thanks to "wintertime")

We have a NetBeans project with updated settings, but it is not at the standard location the IDE expects it at, as the IDE gets slower while a large project, like FreeCol, is open.

You have the option of just clicking on "build.xml" in "Files" pane and starting the build commands easily through the "Navigator" pane inside the IDE that way.

If you opt for using the provided project, it is recommended to use NetBeans 8.1 (or newer), because previous versions contained a bug in that it wont read or write the editor setting for the Java version, if it was correctly set to 1.8 (though it seems version 8.0.2 can read the project file after updating it with 8.1). Just copy the "FreeCol/config/nbproject" folder to "FreeCol/nbproject" once. Without this step it will not work! Open the project inside NetBeans once; it will remember this, as long as you do not close the project manually.

1.8.3 Creating a new NetBeans project (thanks to "xsainnz")

This section is out of date and incomplete. Please, use the existing NetBeans project! Leaving as-is for now in the hope its still educational and updated someday.

- In Netbeans, Select File > New Project
- New Project Window
  - Select Java Category, Java Free-Form Project
- Name and Location Panel
  - In the Location box, browse to where ever you put the source (.../freecol/)
  - It should auto detect the build file location, project name and folder
- Build and Run Actions Panel

  - Leave the settings as they are

- Source Package Folders Panel
  - Add the 'src' folder as Source packages and 'tests' as Test packages
- Java Sources Panel
  - Click 'Add JAR/Folder,
  - browse into the jars folder
  - select all of the jars
  - click open.
- Click Finish

1.9 Code documentation

Our primary code documentation is the Javadoc generated documentation. You can convert this documentation to HTML by typing `ant javadoc`. The directory "javadoc" will then be created and you can start watching the documentation by opening "index.html" from that directory.

There is also some additional documentation here.

1.10 Code Quality

The code you contribute to FreeCol will be read and modified by several different developers. Therefore it is important to create a block of JavaDoc documentation with all methods/classes/packages you implement.

You should also spend more time thinking about the overall structure than when you are working alone.

Please read the Java Code Conventions. This will only take about 15 minutes and will really help you write beautiful code.

Please configure your editor or IDE that code indentations result in the insertion of 4 spaces. Avoid using tabs.

Chapter 2

How to make a FreeCol release

You will need to have installed `ant` to do the build, and `git` to make the commits, however this will be normal for a developer. To generate the online manual you will also need `htlatex`, and `pdflatex` for the print manual. You can avoid these requirements by setting the ant properties `online.manual.is.up.to.date` and `print.manual.is.up.to.date` respectively, however this is not recommended for a release. Uploads to sourceforge use `sftp`.

- Make sure that all relevant changes have been committed to the branch you are about to release. If you plan to upload the manual and/or JavaDoc, regenerate it (with `ant manual` and `ant javadoc`), and fix any JavaDoc errors.

- Merge translations from trunk if the release is not made from trunk, with `ant merge-translations`. Skip this step if the localization files are essentially the same as the ones in trunk (we try to ensure this). Make sure they are up to date, however.

- Lately we release from the git master and continue to work from there, so there is no urgent need to create a special release branch.

- Start a clean compile, run all tests and verify the specification(s). You can do that by calling `ant prepare-commit`.

- Call `ant dist` in order to build all packages. You will be prompted for the version of this release. Alternatively, you can specify the version on

the command line, by with something like: `ant -Dfreecol.version=0.9.0-alpha2 dis` instead (replace 0.9.0-alpha2 with the correct version, of course). It might be necessary to increase the memory available for ant, for example by setting the environment variable `ANT_OPTS="-Xms256m -Xmx256m"`. Errors in language packs only apply to the installer and need not delay the overall release.

- Install one of the generated packages and verify that you can play normally for at least five turns (the java installer can be run from the command line with `java -cp freecol-`*version*`-installer.jar com.izforge.izpack.installer.Installer`). Other good tests include loading a saved game and running the game in debug mode for a hundred turns or so. It might also be a good idea to compile the game from one of the source packages.

- `ant dist` will change the `FREECOL_VERSION` constant in FreeCol.java and the `fcversion` macro in `doc/version.sty` to the release version. Commit these changes.

- Upload the packages to `sftp://frs.sourceforge.net/`. A script `bin/release.sh` is provided that does this job, including rebuilding and uploading the manual and JavaDoc.

- Write a release announcement (see previous versions for comparison). Include information on savegame compatibility with previous FreeCol versions on all messages. Recent practice is to keep the announcement brief but refer to a detailed **Release Notes** page on the sourceforge freecol wiki.

- The source for the `freecol.org` website lives in the git tree in the `www.freecol.org` top-level directory. To update the website, for now, you will need to add a new `www.freecol.org/_posts/freecol-`*version*`-released.html` file and edit the following files:

  `www.freecol.org/download.html` Update the release version number.

  `www.freecol.org/sitemap.html` Add a reference to the new release.

  `www.freecol.org/status.html` Update the release version number.

- Change directory to the `www.freecol.org` subdirectory and build the website with Jekyll (`jekyll build`). This creates a new fully linked version of the website in the `_site` subdirectory.
- Change directory to the `_site` subdirectory and upload to *username*`,freecol@web.sf.net` with `sftp`. The website lives in the `htdocs` subdirectory there. There is a script in `bin/website.sh` script which uploads everything.
- Post the release announcement to:
  - Our mailing lists: developers, translators and users. Beware that you may have to be subscribed to some lists to be able to post. The addresses are: `freecol-{developers,translators,users}@lists.sourceforge.net`
  - The FreeCol forum.
  - The project news page on sourceforge.
- Consider revising the preamble to the "Bugs" page, particularly the "Known Common Problems" section. Go to "Tickets / Admin-Bugs / Options" to make changes.
- Go to the bug tracker and create a new milestone ("Edit Milestones") called "Fixed_*release*" (e.g. "Fixed_0.10.2"). Select the "Fixed_trunk" group and do a mass edit (the pencil icon in the top right of the bug tracker list) and move all bugs to the new milestone. Repeat for the "Pending_Fixed" group setting them to "Closed_Fixed".
- Start a new wiki page for the release notes for the next release.

Chapter 3

Missing features

We know that FreeCol does not yet emulate all features of the original game. However there are several features which are inevitably different to Colonization due to FreeCol being a multiplayer game — interactions with other European players being the obvious example. Similarly we do not attempt to exactly emulate the graphical look and feel of Colonization, although no displayed information should be lost.

Otherwise, any missing feature from Colonization is considered to be a bug. Please report such omissions on the pending features tracker.

FreeCol developers do not have access to the source code of the original Colonization game, so some assumptions are made as to how to implement certain features. This is particularly true in complex, detailed areas such as production and combat. In some cases, players have reverse- engineered the algorithm. If you know of some calcuation in FreeCol that differs from that of the original game, please notify the developer mailing list. There is an effort underway to completely document Colonization's production amounts here.

Chapter 4

Changing the Rules

FreeCol is designed with end-user configuration in mind, so that the game engine can emulate the other similiar games. For this purpose, the developers provide number of configuration options for many of the game's features.

At some point in the future, we will probably add a special rule set editor, but at the moment, the most effective option is to edit the file specification.xml directly. This file defines the abilities of units, founding fathers, buildings, terrain types, goods and equipment, for example. You can find this file in the *data/freecol* directory. Try to avoid overriding the base Colonization-compatible rules in *data/classic*. For a small self-contained rule change, another option is to build a *mod* — see the Mods section following.

This is still work in progress, however, and the schema for the rule set certain to change again in the future. If you wish to develop your own rule set, you will have to monitor FreeCol development closely.

This having been said, we are particularly interested in hearing about problems caused by your changes to the rule set. Some dialogs might be unable to display more types of goods than are currently defined, for example. Or other dialogs might not recognize your new Minuteman unit as an armed unit. Please help us improve FreeCol by telling us about such problems.

If you have a working rule set that adds a new flavor to the game, we will gladly distribute it along with our default rule set. If you have ideas that can not currently be implemented, we will probably try to remove these limitations.

If you try to modify the rule set, you are strongly encouraged to check whether the result is still valid. You can do this by validating the result with the command `ant validate`.

4.1 Modifiers and Abilities

Most of the objects defined by the rule set can be customized via modifiers and abilities. Abilities are usually boolean values ("true" or "false"). If the value is not explicitly stated, it defaults to true. If an ability is not present, it defaults to false. Modifiers define a bonus or penalty to be applied to a numeric value, such as the number of goods produced by a unit. The modifier may be an additive, multiplicative or a percentage modifier. Modifiers default to "identity", which means they have no effect.

The code also checks that all abilities and modifiers it uses are defined by the specification. Therefore, you must define all of them, even if you do not use them. You can do this by setting their value to the default value, e.g. "false" in the case of an ability, or "0" in the case of an additive modifier.

TODO: This section is out of date (version 0.11.2).

model.ability.addTaxToBells
Affects: Player
Provided by: Nation, Nation Type, Founding Father

The player adds the current tax rate as a bonus to bells production. The bonus is modified every time the tax increases or decreases.

model.ability.alwaysOfferedPeace
Affects: Player
Provided by: Nation, Nation Type, Founding Father

The player is always offered peace in negotiations with AI players.

model.ability.ambushBonus
Affects: Unit
Provided by: Nation, Nation Type, Founding Father, Unit Type, Equipment Type

The unit is granted an ambush bonus equal to the terrain's defence value.

model.ability.ambushPenalty
Affects: Unit
Provided by: Nation, Nation Type, Founding Father, Unit Type, Equipment Type

The unit suffers an ambush penalty equal to the terrain's defence value.

model.ability.autoProduction

Affects: Building
Provided by: Building Type

The building needs no units to produce goods, and will never produce more goods than can be stored in the colony.

model.ability.automaticEquipment
Affects: Unit
Provided by: Nation, Nation Type, Founding Father, Unit Type, Equipment Type

The unit automatically picks up equipment if attacked.

model.ability.automaticPromotion
Affects: Unit
Provided by: Nation, Nation Type, Founding Father, Unit Type, Equipment Type

A unit that can be promoted will always be promoted when successful in battle.

model.ability.betterForeignAffairsReport
Affects: Player
Provided by: Nation, Nation Type, Founding Father

The player is provided with more information about foreign powers.

model.ability.bombard
Affects: Unit
Provided by: Nation, Nation Type, Founding Father, Unit Type, Equipment Type

The unit is able to bombard other units.

model.ability.bombardShips
Affects: Building
Provided by: Building Type

The building has the ability to bombard enemy ships on adjacent tiles.

model.ability.bornInColony
Affects: Unit
Provided by: Nation, Nation Type, Founding Father, Unit Type, Equipment Type

The unit can be born in a colony, provided that enough food is available.

model.ability.bornInIndianSettlement
Affects: Unit
Provided by: Nation, Nation Type, Founding Father, Unit Type, Equipment Type

The unit can be born in an Indian settlement, provided that enough food is available.

model.ability.build
Affects: Building
Provided by: Building Type

The building can build units or equipment.

model.ability.buildCustomHouse
Affects: Player
Provided by: Nation, Nation Type, Founding Father

The player can build custom houses.

model.ability.buildFactory
Affects: Player
Provided by: Nation, Nation Type, Founding Father

The player can build factories.

model.ability.canBeCaptured
Affects: Unit
Provided by: Nation, Nation Type, Founding Father, Unit Type, Equipment Type

The unit can be captured. Land units that can not be captured are destroyed, naval units that can not be captured are either sunk or damaged.

model.ability.canBeEquipped
Affects: Unit
Provided by: Nation, Nation Type, Founding Father, Unit Type, Equipment Type

The unit can be equipped.

model.ability.canNotRecruitUnit
Affects: Player

Provided by: Nation, Nation Type, Founding Father

The player can not recruit specified units.

model.ability.captureEquipment
Affects: Unit
Provided by: Nation, Nation Type, Founding Father, Unit Type, Equipment Type

The unit can capture equipment from another unit.

model.ability.captureGoods
Affects: Unit
Provided by: Nation, Nation Type, Founding Father, Unit Type, Equipment Type

The unit can capture goods from another unit.

model.ability.captureUnits
Affects: Unit
Provided by: Nation, Nation Type, Founding Father, Unit Type, Equipment Type

The unit can capture enemy units.

model.ability.carryGoods
Affects: Unit
Provided by: Nation, Nation Type, Founding Father, Unit Type, Equipment Type

The unit can transport goods.

model.ability.carryTreasure
Affects: Unit
Provided by: Nation, Nation Type, Founding Father, Unit Type, Equipment Type

The unit can transport treasures, not treasure trains.

model.ability.carryUnits
Affects: Unit
Provided by: Nation, Nation Type, Founding Father, Unit Type, Equipment Type

The unit can transport other units.

model.ability.convert
Affects: Unit
Provided by: Nation, Nation Type, Founding Father, Unit Type, Equipment Type
The unit is a native convert.

model.ability.dressMissionary
Affects: Building
Provided by: Building Type
The building can commission missionaries.

model.ability.electFoundingFather
Affects: Player
Provided by: Nation, Nation Type, Founding Father
The player can elect Founding Fathers.

model.ability.expertMissionary
Affects: Unit
Provided by: Nation, Nation Type, Founding Father, Unit Type, Equipment Type
The unit is an expert missionary, but not necessarily commissioned.

model.ability.expertPioneer
Affects: Unit
Provided by: Nation, Nation Type, Founding Father, Unit Type, Equipment Type
The unit is an expert pioneer, but not necessarily equipped with tools.

model.ability.expertScout
Affects: Unit
Provided by: Nation, Nation Type, Founding Father, Unit Type, Equipment Type
The unit is an expert scout, but not necessarily equipped with horses.

model.ability.expertSoldier
Affects: Unit
Provided by: Nation, Nation Type, Founding Father, Unit Type, Equipment

Type

The unit is an expert soldier, but not necessarily equipped with muskets.

model.ability.expertsUseConnections
Affects: Player
Provided by: Nation, Nation Type, Founding Father

Experts working in factories can produce a small amount of goods even if the raw materials are not available in the colony.

model.ability.export
Affects: Building
Provided by: Building Type

The building can export goods to Europe directly.

model.ability.foundColony
Affects: Unit
Provided by: Nation, Nation Type, Founding Father, Unit Type, Equipment Type

The unit can found new colonies.

model.ability.foundInLostCity
Affects: Unit
Provided by: Nation, Nation Type, Founding Father, Unit Type, Equipment Type

The unit may be generated as the result of exploring a Lost City Rumour.

model.ability.hasPort
Affects: Colony
Provided by: Map

The colony has access to at least one water tile. This ability can not be set by the specification, but it can be used as a required ability.

model.ability.ignoreEuropeanWars
Affects: Player
Provided by: Nation, Nation Type, Founding Father

The player will not be affected by the Monarch's declarations of war.

model.ability.improveTerrain
Affects: Unit

Provided by: Nation, Nation Type, Founding Father, Unit Type, Equipment Type

The unit is able to improve terrain.

model.ability.independenceDeclared
Affects: Player
Provided by: Nation, Nation Type, Founding Father

The player has declared independence.

model.ability.mercenaryUnit
Affects: Unit
Provided by: Nation, Nation Type, Founding Father, Unit Type, Equipment Type

The unit may be offered as a mercenary unit.

model.ability.missionary
Affects: Unit
Provided by: Nation, Nation Type, Founding Father, Unit Type, Equipment Type

The unit is able to establish missions and incite unrest in native settlements.

model.ability.moveToEurope
Affects: Tile
Provided by: Tile Type

Units on the tile are able to move to Europe.

model.ability.multipleAttacks
Affects: Unit
Provided by: Nation, Nation Type, Founding Father, Unit Type, Equipment Type

The unit can attack more than once.

model.ability.native
Affects: Unit
Provided by: Nation, Nation Type, Founding Father, Unit Type, Equipment Type

The unit is a native unit.

model.ability.navalUnit
Affects: Unit
Provided by: Nation, Nation Type, Founding Father, Unit Type, Equipment Type
The unit is a naval unit.

model.ability.pillageUnprotectedColony
Affects: Unit
Provided by: Nation, Nation Type, Founding Father, Unit Type, Equipment Type
The unit is able to steal goods from and destroy buildings in an unprotected colony.

model.ability.piracy
Affects: Unit
Provided by: Nation, Nation Type, Founding Father, Unit Type, Equipment Type
The unit is a privateer.

model.ability.produceInWater
Affects: Building
Provided by: Building Type
The building enables units to produce on water tiles.

model.ability.refUnit
Affects: Unit
Provided by: Nation, Nation Type, Founding Father, Unit Type, Equipment Type
The unit can be part of the Royal Expeditionary Force.

model.ability.repairUnits
Affects: Building
Provided by: Building Type
The building can repair units.

model.ability.royalExpeditionaryForce
Affects: Player

Provided by: Nation, Nation Type, Founding Father
The player is a Royal Expeditionary Force.

model.ability.rumoursAlwaysPositive
Affects: Player
Provided by: Nation, Nation Type, Founding Father
The player will always get positive results from exploring Lost City Rumours.

model.ability.scoutForeignColony
Affects: Unit
Provided by: Nation, Nation Type, Founding Father, Unit Type, Equipment Type
The unit can scout out foreign colonies.

model.ability.scoutIndianSettlement
Affects: Unit
Provided by: Nation, Nation Type, Founding Father, Unit Type, Equipment Type
The unit can scout out native settlements.

model.ability.selectRecruit
Affects: Player
Provided by: Nation, Nation Type, Founding Father
The player can select a unit to recruit in Europe. This also applies to units generated as a result of finding a Fountain of Youth.

model.ability.teach
Affects: Building
Provided by: Building Type
The building enables experts to teach other units. However, the building may place limits on the experience level of teachers.

model.ability.tradeWithForeignColonies
Affects: Player
Provided by: Nation, Nation Type, Founding Father
The player may trade goods in foreign colonies.

model.ability.undead

Affects: Unit
Provided by: Nation, Nation Type, Founding Father, Unit Type, Equipment Type
The unit is an undead unit (used only in revenge mode).

model.modifier.bombardBonus
Affects: Player
Provided by: Nation, Nation Type, Founding Father
The player's units are granted a bombard bonus when attacking.

model.modifier.buildingPriceBonus
Affects: Player
Provided by: Nation, Nation Type, Founding Father
The player can build or buy buildings at a reduced price.

model.modifier.defence
Affects: Unit
Provided by: Nation, Nation Type, Founding Father, Unit Type, Equipment Type
The unit has a defence bonus or penalty.

model.modifier.immigration *Affects: Player*
Provided by: Goods Type
Goods of this type contribute to the player's immigration points.

model.modifier.landPaymentModifier
Affects: Player
Provided by: Nation, Nation Type, Founding Father
The player can buy Indian land at a reduced price.

model.modifier.liberty *Affects: Player*
Provided by: Goods Type
Goods of this type contribute to the colony's and the owning player's liberty points.

model.modifier.lineOfSightBonus
Affects: Unit
Provided by: Nation, Nation Type, Founding Father, Unit Type, Equipment Type

The unit has an increased line of sight.

model.modifier.minimumColonySize
Affects: Colony
Provided by: Building Type, Nation, Nation Type, Founding Father
The population of the colony can not be voluntarily reduced below this number. The modifier does not in any way affect a population reduction due to starvation or other events.

model.modifier.movementBonus
Affects: Unit
Provided by: Nation, Nation Type, Founding Father, Unit Type, Equipment Type
The unit has an increased movement range.

model.modifier.nativeAlarmModifier
Affects: Player
Provided by: Nation, Nation Type, Founding Father
The player generates less native alarm.

model.modifier.nativeConvertBonus
Affects: Player
Provided by: Nation, Nation Type, Founding Father
The player has a greater chance of converting natives.

model.modifier.nativeTreasureModifier
Affects: Player
Provided by: Nation, Nation Type, Founding Father
The player generates greater treasures when destroying native settlements.

model.modifier.offence
Affects: Unit
Provided by: Nation, Nation Type, Founding Father, Unit Type, Equipment Type
The unit has an offence bonus or penalty.

model.modifier.religiousUnrestBonus
Affects: Player

Provided by: Nation, Nation Type, Founding Father
The player generates greater religious unrest in Europe.

model.modifier.sailHighSeas
Affects: Unit
Provided by: Nation, Nation Type, Founding Father, Unit Type, Equipment Type
The unit's travel time between Europe and the New World is reduced.

model.modifier.tradeBonus
Affects: Player
Provided by: Nation, Nation Type, Founding Father
Prices in the player's market remain stable for longer.

model.modifier.treasureTransportFee
Affects: Player
Provided by: Nation, Nation Type, Founding Father
The player pays a smaller fee for transporting treasures to Europe.

model.modifier.warehouseStorage
Affects: Building
Provided by: Building Type
The building increases the capacity of the warehouse.

Chapter 5

Mods

FreeCol packages a number of simple modifications to the FreeCol rules and resources, generally known as *mods*. The standard mods live in `.../data/mods`. Users can add their own mods to a `mods` directory under their main data directory.

A mod consists of a directory that includes at minimum a file `mod.xml`, a file `FreeColMessages.properties`, and other files that implement the required changes. `mod.xml` simply contains: `<mod id="`*identifier*`" />` where the identifier is some unique name (distinct from other existing mods). The `FreeColMessages.properties` file should contain at minimum entries for `mod.`*identifier*`.name` and `mod.`*identifier*`.shortDescription`, so that the mod selection dialog can display the mod correctly. Other messages needed by the mod belong in `ModMessages.properties`. The difference between these is that `FreeColMessages.properties` is always loaded so that the mod name and description can appear in the mod selection dialog, but `ModMessages.properties` is only loaded if the mod is selected.

Many mods define extra resources. If so, a `resources.properties` file will be needed, and similarly files for the resources involved. For example, the "example" mod defines a "milkmaid" unit, which needs an image, so the example mod directory contains an image file for the milkmaid, and a reference to this file in `resources.properties`.

Most mods change the specification. This is done in a `specification.xml` file. This file is in the same general format as the existing freecol rule sets, but does not attempt to provide a comprehensive set. The first non-comment line should be: `<freecol-specification id="`*identifier*`">`. Note that mods typically do not specify which ruleset they extend.

When modifying a specification, expect additional elements you provide to be applied additively, but attributes of an existing element are cleared unless a special `preserve="true"` attribute is present. If you need to delete or redefine an element, respecify it in context with just its `id` and `delete="true"` attributes. So for example, in the freecol ruleset we need to remove the `model.modifier.minimumColonySize` modifier from the stockade building, which is done as follows:

```
<building-type id="model.building.stockade" preserve="true">
  <modifier id="model.modifier.minimumColonySize" delete="true" />
</building-type>
```

Note that not all elements take a `delete` tag yet. You may need to read or modify the source to be certain a mod will work.

Chapter 6

Resources

Various links pointing to more or less reliable information about the original Colonization game:

- Strategy Wiki
- The Unofficial Microprose Colonization Home Page
- Play Colonization at Viceroy level
- Colonization Fan Page
- Bill Cranston's Strategy guide
- Tomasz Wegrzanowski' Strategy Guide, contains very valuable material on the number of bells required to elect a Founding Father, among other things

Chapter 7

Map Format

A map file, as found in `.../data/maps`, is a cut-down version of a normal freecol save file. Map files should have the `.fsm` ("FreeCol Saved Map") extension, as distinct from `.fsg` ("FreeCol Saved Game"). Saved maps and games are both actually just standard Java archives, as operated on by the `jar(1)` utility.

Saved maps should contain the following files:

- `savegame.properties`, a standard Java properties file containing the map.height and map.width properties.
- `thumbnail.jpg`, a high level picture of the map, as used in the map selection dialog
- `savegame.xml`, the XML representation of a FreeCol game, but with the following skeleton:
  - `<?xml>` header
  - `savedGame`
    - `game`
      - `cibola` (seven entries)
      - `nationOptions` (native `nationOption`s only)
      - `player` (native players only)
      - `map`