www.suse.com

Open Build Service

Open Build Service: Reference Guide

List of Authors: Adrian Schröter

Copyright © 2006–2012 Novell, Inc. and contributors. All rights reserved.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or (at your option) version 1.3; with the Invariant Section being this copyright notice and license. A copy of the license version 1.2 is included in the section entitled “GNU Free Documentation License”.

For Novell trademarks, see the Novell Trademark and Service Mark list http://www.novell.com/company/legal/trademarks/tmlist.html . Linux* is a registered trademark of Linus Torvalds. All other third party trademarks are the property of their respective owners. A trademark symbol (®, ™ etc.) denotes a Novell trademark; an asterisk (*) denotes a third party trademark.

All information found in this book has been compiled with utmost attention to detail. However, this does not guarantee complete accuracy. Neither Novell, Inc., SUSE LINUX Products GmbH, the authors, nor the translators shall be held liable for possible errors or the consequences thereof.

Contents

About this Guide

This book is part of the official Open Build Service documentation. These books are considered to be reviewed and edited to be the reference documentation of OBS.

These books are not considered to be focused on a special OBS version. They are also not a replacement of the documentation inside of the openSUSE wiki. But content from the wiki may get consolidated and included in these books.

Furthermore these books get written by the OBS community, please check the chapter how to work on these books. We request esp. experienced users and administrators to join and to contribute to these books. It is not required to be a very good or even native English speaker, because we rely on community editors to improve the language.

1

OBS Motivation

An overview about OBS history and motivation to build it.

1.1 Pre OBS times

SUSE built all distributions with an inhouse developed autobuild system. It was designed to collect source ....

1.1.1 Initial Goals for OBS

...

1.1.2 OBS history

...

1.2 Current Motivation of OBS

...

1.3 Future dreams

...

2 OBS Concepts

..

2.1 Project organization

All sources and binaries which are hosted inside of OBS are organized in projects.

2.1.1 Project meta data

A project gets configured in the project `/source/$PROJECT/_meta` path. It can be edited in web interface in the `RAW Config` tab or via command line via

```
osc meta prj -e $PROJECT
```

This file contains

- generic description data in `title` and `description` elements.
- An ACL list of users and groups connected with a role. The `maintainer` role defines the list of users permitted to commit changes to the project.
- A number of flags, controlling the build and publishing process and possible read access protections.
- A list of repositories to be created. This list defines what other repositories should be used, which architectures shall be built and build job scheduling parameters.

The following flags can be used to control the behavior of a package or project. Most of them can also be limited to specified repositories or architectures.

- `build` flag defines if package sources should get build. If enabled, it signals the scheduler to trigger server side builds based on events like source changes, changes of packages used in the build environment or manual rebuild triggers. A local build via CLI is possible independent of this flag. Default is enabled.

- `publish` can be used to enable or disable publishing the build result as repository. This happens after an entire repository has finished build for an architecture. A publish also gets triggered, when the publish flag is enabled after a repository finished the build. Default is enabled.

- `debuginfo` can be used to modify the build process to create debuginfo data along with the package build for later debugging purposes. A flag change is not triggering rebuilds, it just affects the next build. Default is disabled.

- `useforbuild` is used to control if a built result shall be copied to the build pool. This means it will get used for other builds in their buildenvironment. When this is disabled, the build has no influence on builds of other packages using this repository. In case a former build exists the old binaries will be used. Disabling this flag also means that "wipe" commands to remove binary files will have no effect on the build pool. A flag change is not triggering rebuilds, it just affects the next build. Default is enabled.

- `access` flag can be used to hide an entire project. This includes binaries and sources. It can only be used at project creation time and can just be enabled (making it public again) afterwards. This flag can only be used on projects. Default is enabled.

- `sourceaccess` flag can be used to hide the sources, but still show the existence of a project or package. This also includes debug packages in case the distribution is supporting this correctly. This flag can only be used at package creation time. There is no code yet which is checking for possible references to this package. Default is enabled.

- `downloadbinary` permission still exists like before. However, unlike "access" and "sourceaccess" this is not a security feature. It is just a convenience feature, which makes it impossible to get the binaries via the API directly. But it is still possible to get the binaries via build time in any case. Default is enabled.

2.1.2 Project build configuration.

A project gets configured in the project `/source/$PROJECT/_config` path. It can be edited in web interface in the `Project Config` tab or via one of the following command lines

```
osc meta prjconf -e $PROJECT
osc co $PROJECT _project
```

This file contains information on how to setup a build environment.

2.1.3 Project build macro configuration

The macro configuration is part of the build configuration in `/source/$PROJECT/_config`. It can be added at the end behind a `Macro:`

2.1.4 An OBS Package

An OBS Package is a sub name space below a project. It contains the specification of a single package build for all specified repositories.

2.2 The API

...

2.3 The OBS Interconnect

...

2.4 Attribute System

...

2.5 Automatic source processing

...

3

Build Process

Each package build is done in a fresh environment. This is done to ensure that no dependencies are missing and every later build produces identical results.

3.1 How is a build process defined

...

3.2 How does a build process work

All sources and binaries which are hosted inside of OBS are organized in projects.

3.2.1 Preinstall phase

...

3.2.2 Install Phase

..

3.2.3 Package build

..

3.2.4 Post build steps

..

3.3 Different ways to build

...

3.4 Security aspects

...

4 Source Management

..

4.1 Find Package Sources

OBS is adding information to each created package about the origin of the sources. This information is stored in the `DISTURL` tag of an rpm, which can be displayed as follows

```
rpm -q --queryformat '%{DISTURL}\n' glibc
```

```
rpm -q --queryformat '%{DISTURL}\n' -p glibc-2.1.0-1.i586.rpm
```

The disturl can look like this: `obs://build.opensuse.org/openSUSE:Factory/standard/80d21fdd2299302358246d757b4d8c4f-glibc` It always starts with obs://. The second part is the name of the build instance, which usually also hosts the webui. Next comes the project name and the repository name where the binary got built. Last part is the source md5 sum and the package name.

The disturl can also be entered in the search field of the web interface of the build service.

The Request And Review System

5

..

6 Package Formats

OBS is written to not be package format specific. However, every new format needs support in OBS. So far it does support rpm, debian, kiwi appliances and products.

6.1 Setup a format

The default format in OBS is rpm, other formats need to be specified in prjconf.

6.2 Format Specials

Each format has some specials

6.2.1 rpm

For building rpms you need a .spec file for each package, containing its build description. ...

6.2.2 deb

...

6.2.3 pkg

Archlinux...

6.2.4 kiwi appliance

...

6.2.5 kiwi product

...

7

Build Configuration

The build configuration is needed at least in each base project to define the setup of the build system. In addition to that it can be used to handle compatibility layers or to switch on or off certain features during the build.

The build config is reachable via API /source/PROJECT/_config, via `osc meta prjconf` or via webui via Project Config tab.

7.1 Setup a format

The default format in OBS is rpm, other formats need to be specified in prjconf.

7.2 Keywords

The build configuration is parsed by OBS in rpm style independend of the used packaging format.

7.2.1 Required: <PACKAGES>

Required lines contain one or more packages which are needed always to start the build. When these packages do change a new build of all packages get triggered automatically.

7.2.2 Support: <PACKAGES>

Support lines contain one or more packages which are needed always to start the build. However it is considered that changes in these packages do not affect other builds, therefore no builds get triggered automatically.

7.2.3 Preinstall: <PACKAGES>

Preinstall packages are needed to run the package installation tool. They get unpacked before the VM gets started. Included scripts are NOT executed during this phase. However these packages will get installed again inside of the VM including script execution.

7.2.4 Keep: <PACKAGES>

The building package does not get installed by default, even when it is required. Keep is overwriting this behaviour and lets the old build installed. It is usually needed for most preinstalled packages and critical required packages at least.

7.2.5 VMInstall: <PACKAGES>

VMInstall is like Preinstall, but these packages get only installed when XEN or KVM is used.

7.2.6 Runscripts: <PACKAGES>

Runscripts defines the scripts of preinstalled packages which needs to be executed directly after preinstall phase, before installing the remaining packages.

7.2.7 Order: <PACKAGE_A>:<PACKAGE_B>

The build script takes care about the installation order if they are defined via PreRequires inside of the packages. However, there might be dependency loops (reported during setup of the build system). The Order statement can be used then to give a hint where to break the loop. <PACKAGE_A> will get installed before <PACKAGE_B>.

7.2.8 ExportFilter: <GLOB> <ARCHITECTURES>

The export filter can be used to export build results from one architecture to others. This is required when one architecture needs packages from another one to build against. The glob must match the resulting binary name of the package. It will export it to all listed scheduler architectures. Using a single dot will export it to the architecture which was used to build it. So not using a dot there will filter the package.

7.2.9 Prefer: <PACKAGES>

In case multiple packages could be installed to satisfy a dependency, the OBS server will complain about that situation (Unlike like most standard package installation tools, which just pick one). This is done to avoid the situation that builds are not reproducable since random packages would get used. The Prefer: tag lists packages to be prefered in case a choice exists. When the package name is prefixed with a dash, this is a de-prefer.

7.2.10 Prefer: <PACKAGE_A>:<PACKAGES>

It is possible to define the prefer only when one package is creating the choice error. This package must be listed first with a colon.

7.2.11 Ignore: <PACKAGES>

Ignore can be used to override dependencies. This can be usefull to break dependency cycles in cases when a package dependency is only required in installed systems, but not required to build against this package.

7.2.12 Ignore: <PACKAGE_A>:<PACKAGES>

It is possible to define the ignore only for one package. This package must be listed first with a colon.

7.2.13 FileProvides: <FILE> <PACKAGE>

Required dependendencies to files (instead of package names) are ignore by OBS by default. FileProvides can be used to map a file to a certain package name. File needs to have the full path. Keep in mind that some publish repository formats do not export filenames or only some of them. So installing these packages may not be possible for the user.

7.2.14 Substitute: <PACKAGE_A> ;<PACKAGES>

It is possible to replace to BuildRequires with other packages. This will have only an affect on direct BuildRequired packages, not on indirect required packages.

7.2.15 Optflags: <TARGET_ARCHITECTURE> <FLAGS>

rpm only: Optflags exports compiler flags to the build. They will only have an effect when the spec file is using $RPM_OPT_FLAGS. The target architecture may be * to affect all architectures.

7.2.16 Target: <TARGET_ARCHITECTURE>

rpm only: Defines the target architecture. This can be used to build for i686 on i586 schedulers for example.

7.2.17 HostArch: <HOST_ARCHITECTURE>

This is used for cross builds via qemu emulation. It defines the host architecture, while the scheduler architecture remains the target architecture by default.

7.2.18 Repotype: <TYPE> <HINT>

Defines the repository format. Valid values are: none, rpm-md(default), rpm-md:deltainfo, rpm-md-legacy, suse, debian, hdlist2, comps. To split the debug packages in an own published repository the hint splitdebug:$REPOSIORY_SUFFIX can be used.

7.2.19 Binarytype: <TYPE>

(OBS 2.4 or later):.

7.2.20 Patterntype: rpm-md ymp: <TYPES>

Defines the pattern format. Valid values are: rpm-md(default), ymp.

7.2.21 Conditions

The following conditions can be used to modify the config.

%define <KEY> <VALUE>

Defines a key with a value. This value can be used in the entire build config by using %{KEY}

%if ... %else ... %endif

Lines can become conditional using the rpm syntax. It may test one keys set by define or predefined keys (%_project or %_repository).

%ifarch <SCHEDULER_ARCHS> ... %else ... %endif

Lines can become conditional using the rpm syntax.

%ifnarch <SCHEDULER_ARCHS> ... %else ... %endif

Lines can become conditional using the rpm syntax.

7.3 Macro Section

The macro section of the build configuration starts behind a Macros: line and is only used on rpm builds. The content gets exported to be used by rpmbuild during the build. It can be used to export a define using "%key value" lines.

8

Source Services

Source Services are tools to validate, generate or modify sources in a trustable way. They are designed as smallest possible tools and can be combined following the powerful idea of the classic UNIX design.

Design goals of source services were:

- server side generated files must be easy to identify and must not be modifiable by the user. This way others user can trust them to be generated in the documented way without modifications.
- generated files must never create merge conflicts
- generated files must be a separate commit to the user change
- services must be runnable at any time without user commit
- services must be runnable on server and client side in the same way
- services must be designed in a safe way. A source checkout and service run must never harm the system of a user.
- services shall be designed in a way to avoid unnecessary commits. This means there shall be no time-dependent changes. In case the package already contains the same file, the newly generated file must be dropped.
- local services can be added and used by everybody.
- server side services must be installed by the admin of the OBS server.

- service can be defined per package or project wide.

8.1 Using services for validation

Source Services may be used to validate sources. This can happen per package, which is useful when the packager wants to validate that downloaded sources are really from the original maintainer. Or validation can happen for an entire project to apply general policies. These services can't get skipped in any package

Validation can happen by validating files (for example using the `verify_file` or `source_validator` service. These services just fail in the error case which leads to the build state "broken". Or validation can happen by redoing a certain action and store the result as new file as `download_files` is doing. In this case the newly generated file will be used instead of the committed one during build.

8.2 Different Modes when using services

Each service can be used in a special mode defining when it should run and how to use the result. This can be done per package or globally for an entire project.

8.2.1 Default Mode

The default mode of a service is to always run after each commit on the server side and locally before every local build.

8.2.2 trylocal Mode

The trylocal mode is running the service locally when using current osc versions. The result gets committed as standard files and not named with _service: prefix. Additionally the service runs on the server by default, but usually the service should detect that the result is the same and skip the generated files. In case they differ for any reason (because the webui or api was used for example) they get generated and added on the server.

8.2.3 localonly Mode

The localonly mode is running the service locally when using current osc versions. The result gets committed as standard files and not named with _service: prefix. The service is never running on the server side. It is also not possible to trigger it manually.

8.2.4 disabled Mode

The disabled mode is neither running the service locally or on the server side. It can be used to temporarily disable the service but keeping the definition as part of the service definition. Or it can be used to define the way how to generate the sources and doing so by manually calling

```
osc service disabledrun
```

The result will get committed as standard files again.

8.3 How are source service definitions stored

The called services are always defined in a `_service` file. It is either part of the package sources or used project-wide when stored inside the `_project` package.

The _service file contains a list of services which get called in this order. Each service may define a list of parameters and a mode. The project wide services get called after the per package defined services. The _service file is an xml file like this example:

```
<services>
  <service name="download_files" mode="trylocal" />
  <service name="verify_file">
    <param name="file">krabber-1.0.tar.gz</param>
    <param name="verifier">sha256</param>
    <param
 name="checksum">7f535a96a834b31ba2201a90c4d365990785dead92be02d4cf846713be938b78</
param>
  </service>
  <service name="update_source" mode="disabled" />
</services>
```

This example downloads the files via download_files service via the given URLs from the spec file. When using osc this file gets committed as part of the commit. After-

wards the krabber-1.0.tar.gz file will always be compared with the sha256 checksum. And last but not least there is the `update_source` service mentioned, which is usually not executed. Except when `osc service disabledrun` is called, which will try to upgrade the package to a newer source version available online.

8.4 How to write a source service

..

8.5 Interfaces for using source services

..

9

Signing

OBS can sign build results.

9.1 Package signing

Each package is signed with a PGP key to allow checking its integrity on user's machines.

9.2 Repository signing

...

9.3 Product signing

...

9.4 Configure sign key

...

10 Product Building

OBS is used to build complete product lines. It is employed for openSUSE and also for SUSE Linux Enterprise products.

10.1 Requirements of a product

...

10.2 Possible shapes of a product

product media, appliance, repository...

10.3 Product Setup in OBS

....

10.3.1 Package building

A project is configured in the project meta data. This meta data contains ... These projects contain ...

10.3.2 Appliance setup with kiwi

...

10.3.3 Product creation with kiwi

...

10.3.4 Product Line Setup

Bigger products, like openSUSE or SUSE Linux Enterprise usually contain multiple media. These media must be kept in sync in various ways, for example for installation patterns, version numbering and other meta data. To guarantee this, OBS provides the product definition xml. Submitting such a product xml will create all necessary sources for meta packages and also all needed kiwi source files.

10.4 Release Management

...

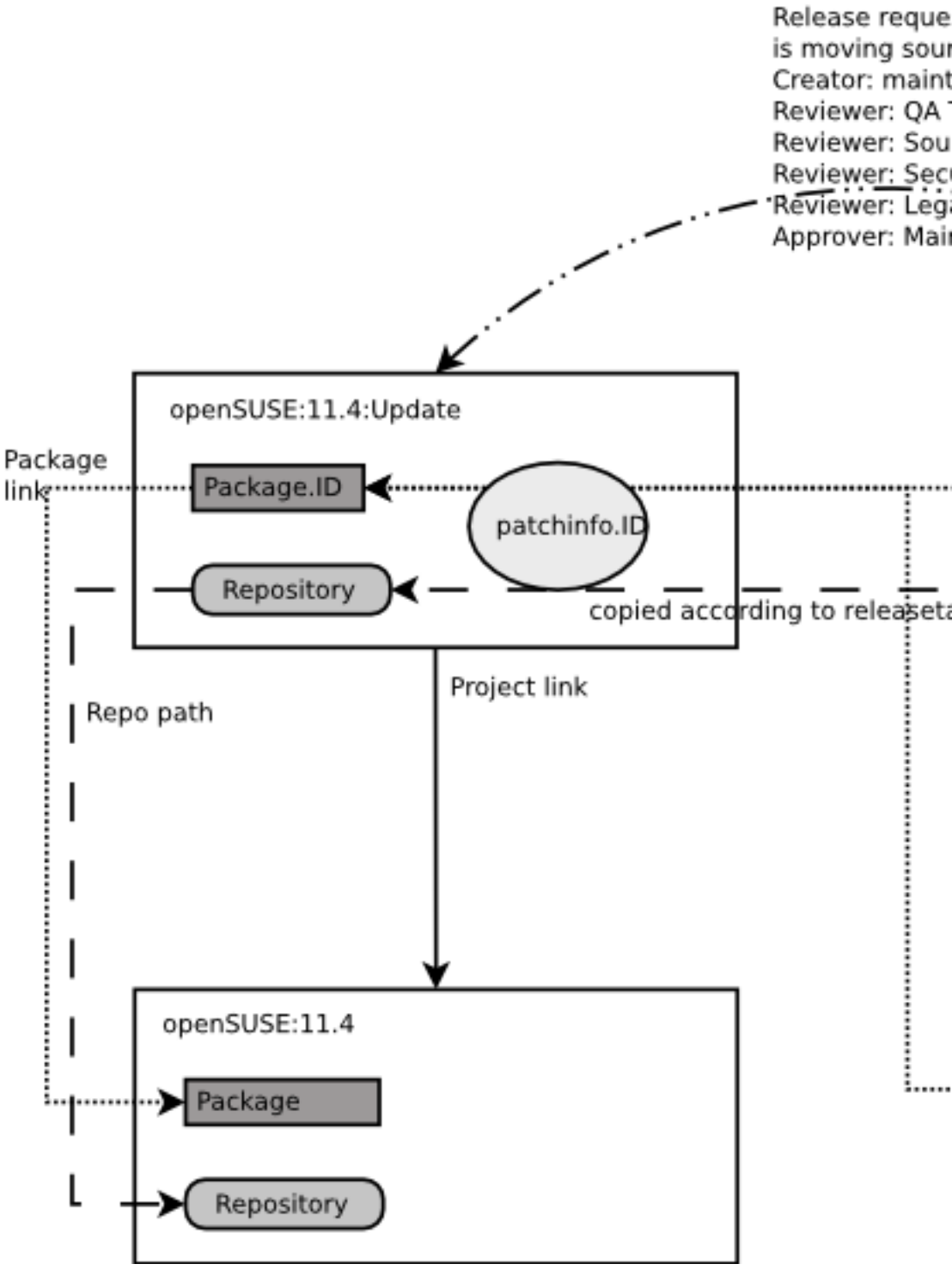
10.5 Typical Review Process Setup

...

11 The Maintenance Support

This chapter explains the setup and workflow of an maintenance update in the openSUSE way. However, this should not be limited to openSUSE distribution projects but be usable anywhere (the entire workflow or just parts of it).

The goal of the OBS maintenance process is to publish updates for a frozen project, in this example an entire distribution. These updates need to be approved by a maintenance team and the published result must contain documentation about the changes and be applicable in the easiest way by the users. The result is a package repository with additional informations about the solved problems and defined groups of packages to achieve that. Also binary delta data may get generated to reduce the needed download size for the clients.


Release reque
is moving sour
Creator: maint
Reviewer: QA
Reviewer: Sou
Reviewer: Sec
Reviewer: Lega
Approver: Mai
openSUSE:11.4:Update
Package
link
Package.ID
patchinfo.ID
Repository
copied according to releaseta
Project link
Repo path
openSUSE:11.4
Package
Repository

This figure is giving a basic overview about the project setup and general workflow for a single package and single maintained distribution. It shows the "openSUSE:11.4" project, which is considered to be frozen and not changing at all anymore. The "openSUSE:11.4:Update" projects hosts all officially released updates. It does not build any binary, just gets it sources and binaries from the maintenance incident project via the release process. The incident project is named "openSUSE:Maintenance:IDxxx" in this example, which is under control of the maintenance team. Official updates get built and reviewed here. QA teams are also testing the binaries from here. However, a user can prepare it in the same way in his project and start the maintenance process via doing a "maintenance" request.

- openSUSE:11.4 is the GA Project (page 47) in this example. Frozen and not changing anymore.
- openSUSE:11.4:Update is the Update Project (page 48) to release official updates.
- openSUSE:Maintenance is the Maintenance Project (page 48).
- openSUSE:Maintenance:IDxxx is the Incident (page 48) project.

11.2 How to use the maintenance process

This describes all required steps by all involved persons from preparing to releasing a maintenance update.

11.2.1 Way A: A maintainer builds an entire update incident for submission

A user is usually starting to prepare an update by creating a maintenance branch. This is typically done by creating an own maintenance project. Usually multiple released products are affected, so the server can find out which one are maintained by a given source package name, in this example for glibc including checkout via

```
osc mbranch glibc
osc mbranch --checkout glibc
```

This is equivalent to the api call `/source?cmd=branch&package=glibc`.

It is also possible to branch only one defined version, if it is known that only one version is affected. In this example the openSUSE:12.1 version:

```
osc branch --maintenance openSUSE:12.1 glibc
osc branch -M -c openSUSE:12.1 glibc
```

NOTE: both branch commands do support the --noaccess parameter, which will create a hidden project. This may be used when a not yet public known security shall get fixed.

Afterwards the user needs to do the needed modifications. Packages will be built and can be tested. Afterwards he may add informations about the purpose of this maintenance update via

```
osc patchinfo
```

If the source changes contain references to issue trackers (like bugzilla, CVE or FATE) these will be added to the _patchinfo file.

The server will create a full maintenance channel now, in case the user wants to test this as well. After the user has tested, he has to create a maintenancerequest to ask the maintenance team to accept this as official update incident:

```
osc maintenancerequest
```

On accepting this request all sources of the entire project will get copied to the incident project and be rebuild. The origin project gets usually removed (based on the request cleanup options).

11.2.2 Way B: Submitting a package without branching

You may submit a package source from a project which is not prepared as maintenance project. That works via the maintenancerequest mechanism by specifing one or more packages from one project. As a consequence it means also that the first testable build will happen in the maintenance incident project. Also the maintenance team need to write the update information on their own.

```
            osc maintenancerequest [ SOURCEPROJECT [ SOURCEPACKAGES
 RELEASEPROJECT ] ]
```

The following example is submitting two packages (kdelibs4 and kdebase4) from the project KDE:Devel project as update for openSUSE:12.1

```
osc maintenancerequest KDE:Devel kdelibs4 kdebase4 openSUSE:12.1
```

NOTE: it is also possible to specify an existing incident as target via the --incident parameter. The packages will be merged into that existing incident project then.

11.2.3 Way C: Process gets initiated by the maintenance team

The maintenance team may start the process (for example because a security issue was reported and the maintenance team decided that a fix is required). In this case the incident gets created via the webui or via the api call

```
osc createincidient [PROJECT]
```

```
osc api /source/PROJECT?cmd=createmaintenanceincident
```

```
osc api /source?
cmd=createmaintenanceincident&attribute=OBS:Maintenance.
```

To document the expected work the creation of a patchinfo package is needed. This can be done via

```
osc patchinfo [PROJECT]
```

It is important to add bugzilla entries inside of the _patchinfo file. As long these are open bugzilla entries, the bug assignee will see this patchinfoi on his "my work" webui and osc views, so he knows that work is expected for him.

11.2.4 Maintenance Incident Processing

The maintenance incidents are usually managed by a maintenance team. In case the incident got started by a maintainer a maintenance request is targeted towards the defined maintenance project, in our example this is openSUSE:Maintenance. The de-

fined maintainer and reviewers in this project need to decide about this request. In case it gets accepted, the server is creating a subproject with a unique incident ID and copies the sources and build settings to it. The origin project will get removed usually via the cleanup option. This maintenance project is used to build the final packages.

If the maintenance team decides to merge a new maintenance request with an existing incident, they can run the `osc rq setincident $REQUESTID $INCIDENT` before accepting the request.

The maintenance team may still modify them or the patchinfo data at this point. An outside maintainer can still submit changes via standard submit request mechanism, but direct write permissions are not granted. When the maintenance people are satisfied with the update, they can create a request to release the sources and binaries to the final openSUSE:11.4:Update project.

```
osc releaserequest
```

The release request needs to specify the source and target for each package. In case just the source package or project is specified the api is completing the request on creation time. It is using this based on the source link target of each package and the release information in the repository definitions.

11.2.5 Incident gets released

The release process gets usually started via creating a release request. This sets all affected packages to the locked state, which means that all commands for editing the source or triggering rebuilds are not allowed anymore.

The release request typically needs to be approved by QA and other teams as defined in the Update project. In case something gets declined, the necessary changes need to be submitted to the maintenance project and a new release request has to be created.

A unique release ID will be generated and become part of the updateinfo.xml file in the target project on release event. This ID is different from the incident ID and is usually in the style of "YEAR-COUNTER". The counter is strictly increasing on each release. In case of a re-release of the same incident a release counter will be added.

A different naming scheme can be defined via the OBS:MaintenanceIdTemplate attribute value. The release will move all packages to the update project and extend the target package name with the incident ID. Binaries will be moved as well without

modification. The exception is the updateinfo.xml which will be modified by replacing its incident id with the release id.

11.2.6 Incident gets reopened and re-released

An update should not, but may have an undetected regression. In this case the update needs a re-release. (If another problem shall be fixed a new incident should be created instead.)

If the current update harms the systems, the maintenance team may decide to take it back immediatly. It can be done by removing the patchinfo.ID package container in the Update projects. This will create a new update channel without this update.

To re-open a release incident project, it must get unlocked and marked as open again. Unlocking can be done either via revoking a release request or via explicit unlocking the incident. The explicit unlock via osc:

```
osc unlock INCIDENT_PROJECT
```

is also triggering a rebuild to ensure to have higher release numbers and adding the "trigger=maintenance" flags to the release target definitions. Afterwards the project can be edited again and also gets listed as running incident again.

11.2.7 Using custom update IDs

The used string of update IDs can be defined via the OBS:MaintenanceIdTemplate attribute value of the master maintenance project.

11.3 OBS Internal Mechanisms

OBS is tracking maintenance work and can be used as a database for future and past updates.

11.3.1 The workflow step by step

A maintenance incident is started by creating the incident project, either via a developer request or by the maintenance team.

1. Incident project container is created. This is always a sub project to the maintenance project. A unique ID (counter) is used as subproject name. Build is disabled by default project wide.

2. Default content for an incident is added via branch by attribute call:

   - Package sources get added based on given package and attribute name from all existing project instances. The package name is extended by the source project name to allow multiple instances of same package in one project. Source revision links are using the xsrcmd5 to avoid that other releases will affect this package instance. TBD: how to raise the source revision to current version manually ?

   - Build repositories are added if missing. All repositories from all projects where the package sources gets branched from are used. The build flags in the package instances gets switched on for these.

   - A release target definition is added to the repository configuration via additional releasetarget element. The special release condition "maintenance" gets defined for this.

3. Fixes for the packages need to get submitted now.

4. A patchinfo file need to get added describing the issue.

5. OBS server is building packages according to the sources and update information according to the patchinfo data.

6. one or more release requests get created. It does also set the project to "freeze" state by default, this means no source changes are possible anymore and all running builds get canceled.

7. Usually the request is in review state with defined reviewers from the release project. All reviewers need to review the state in the incident project.

8. Request changes into state "new" when all reviewers accepted the release request.

9. The release happens on accepting the request by the maintainers of the release project.

   - All package sources and binaries get copied into a package container where the package name gets extended by the incident number.

- A main package gets created or updated, it just contains a link to the current incident package. Eg glibc points to glibc.42. The purpose of this main package is to have a place to refer to the current sources of a package.
- The release target condition=maintenance gets removed.
- The updateinfo.xml gets updated with the existing or now created unique updateinfo ID.
- The server will update the repository based on all existing binaries.

10. OPTIONAL: A maintenance coordinator may remove the release by removing the package instances inside the release project. The source link has to be fixed manually. (We may offer a function for this).

11. OPTIONAL: A maintenance incident can be restarted by

- Removing the lock flag.
- Adding again the condition=maintenance attribute to the release target which requires a re-release.

NOTE: The step 1 and 2 may be done via accepting an incident request instead.

11.3.2 Search for incidents

The webui shows the running and past incidents when going to the maintenance project (openSUSE:Maintenance in our example). It shows the open requests either for creating or release an incident. Also the open incidents, which are not yet released are visible.

All users need usually just to visit their "my work" screen in webui or osc to see requests or patchinfog were actions of them are expected:

```
osc my [work]
```

The following items list some common ways to search for maintenance incidents via the api:

- A developer can see the work to be done by him via searching for patchinfos with open bugzilla entries:

```
/search/package?match=([kind='patchinfo' and issue/[@state='OPEN' and
 owner/@login='$USER_LOGIN']])
```

- A maintenance coordinator can see requests for doing a maintenance release via searching for open requests with maintenance_incident action against the maintenance project. They are visible in the webui request page of that project or via

```
/search/request?match=(state/@name='new') and action/
@type='maintenance_incident' and action/target/
@project='openSUSE:Maintenance')
```

- A maintenance coordinator can see open incidents via searching for incidents project repositories which have a release target with maintenance trigger. Note: this search result is showing all repositories of a matching project.

```
/search/project?match=(repository/releasetarget/@trigger='maintenance')
```

- A maintenance coordinator can see updates which currently are reviewed (for example by a QA team) via

```
/search/request?match=(state/@name='review') and action/
@type='maintenance_release')
```

- A maintenance coordinator can see updates ready to release via searching for open requests with maintenance_release action.

```
/search/request?match=(state/@name='new') and action/
@type='maintenance_release')
```

11.4 How to setup projects for doing a maintenance cycle

11.4.1 Defining a maintenance space

An OBS server is using by default a maintenance space defined via the OBS:Maintenance attribute. This must get created on a project where maintenance incident projects should get created below. This project is also defining the default maintenance maintainers and reviewers in it's acl list.

It is possible to have multiple and independent maintenance name spaces, however the maintenance request must be created against this other namespace manually or using a different attribute.

11.4.2 Maintained Project Setups

Maintained projects must be frozen, this means no changes in sources or binaries. All updates will be hosted in the defined update project. This project gets defined via the OBS:UpdateProject attribute which must contain a value with the update project name. In addition to this, an attribute to define the active maintenance should also be defined, by default the OBS:Maintained attribute. The osc mbranch command will take packages from this project as a result.

The Update project should be defined as build disabled as well. Also define a project link to the main project and at least one repository building against the main project.

12

Cross Architecture Build

OBS can build for architectures without having the hardware. This is done via emulation steps. However, this is not a classic build with cross tool chains. The advantage is that the build environments of the packages don't need support for cross tools.

13 Administration

This chapter describes the components of an OBS server and the typical administration tasks for an OBS administrator.

This chapter is not intended to describe special installation hints for a certain OBS version, please refer to the online resources or the README.SETUP file for this.

13.1 Server Components

To be written...

13.2 Tools for the admin

To be written...

13.2.1 obs_admin

To be written...

13.3 Integrate OBS into your environment

To be written...

13.3.1 Integrate Notifications

To be written...

13.3.2 Integrate your user management in OBS

To be written...

14 openSUSE Factory

This chapter describes how the development of the future openSUSE distribution is done within OBS.

14.1 openSUSE:Factory project

The main project is openSUSE:Factory. This project is controlled by a small group which does review all submissions according to the policies. Submissions are possible via submit requests, which get reviewed by default by two groups: The Legal team and the code review team. ...

14.2 Devel Projects

The goal of openSUSE:Factory is to always have a working state. This is needed to allow all developer groups to use it as a base for testing their own, possibly experimental work in their own projects. To have a complete test of ...

Glossary

Open Build Service
: The Open Build Service and its acronym OBS is used to speak about the server part of the build service. When speaking about OBS all possible instances are affected.

openSUSE Build Service
: The openSUSE Build Service is the concrete instance of Open Build Service (page 47) from the openSUSE project at http://build.opensuse.org.

Appliance
: A software appliance is a preconfigured combination of an application (for example, a Web server) and its configuration, and includes an operating system (for example, SUSE Linux Enterprise Server). All these parts are integrated into a single image and can be deployed on industry hardware or on a virtual environment.

EULA
: End User License Agreement. For software that needs a special license (usually non-open source) which the user has to agree to before installing.

KIWI
: KIWI provides a complete operating system image solution. It can create images for Linux supported hardware platforms or for virtualization systems.

Overlay Files
: Files which are created, removed, or modified in your testdrive are considered as *overlay files*. These files can be added later as a supplement to your appliance.

Binaries
: Binaries are considered as build results of OBS Projects. Binaries can be reused in an environment to build further binaries. Currently OBS is supporting rpm, deb and all formats generated by KIWI (page 47).

GA Project
: The GA project builds an initial release of a product. It gets frozen after releasing the product. All further updates get released via the Update Project (page 48) of this project.

Update Project

The update project is a Release Project (page 48) which provides official updates for the products generated in the GA Project (page 47). The Update project is usually linking (sources and repositories) against the GA Project (page 47).

Maintenance Project

The maintenance project is a project without sources and binaries, defined by the maintenance team. Incident (page 48)s are created as sub projects of this project.

Incident

The maintenance incident describes a concrete problem and the required updates. If the problem exists for multiple code streams, one incident covers all of them. An incident is started by creating a maintenance incident project and the update get built here.

Release Project

A release project is hosting a release repository which is not building any packages ever. It is just used to copy sources and binaries to this project on a release event.

A

How to work on this Book

These books are written with docbook and can be converted to html or pdf documentation. Please use the following command to checkout the current source of this book:

```
svn co https://svn.opensuse.org/svn/opensuse-doc/trunk/documents/obs/en OBS-documentation
```

Please check the README file for descriptions how to validate and generate them.

We ask every experienced OBS user or admin to join to work on these books. Subversion commit permissions will be granted after the first submitted patch. It is even possible to host instance specific content in the official subversion repository, it is just a matter to tag them correctly. Special parts of this documentation are tagged as `<para os="opensuse;meego">` for example. In this case the paragraph will become only visible when creating the openSUSE or MeeGo book.

B

GNU Licenses

This appendix contains the GNU General Public License version 2 and the GNU Free Documentation License version 1.2.

GNU General Public License

Version 2, June 1991

Copyright (C) 1989, 1991 Free Software Foundation, Inc. 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

GNU GENERAL PUBLIC LICENSE TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The “Program”, below, refers to any such program or work, and a “work based on the Program” means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term “modification”.) Each licensee is addressed as “you”.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program’s source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
one line to give the program's name and an idea of what it does.
Copyright (C) yyyy name of author


This program is free software; you can redistribute it and/or
modify it under the terms of the GNU General Public License
as published by the Free Software Foundation; either version 2
of the License, or (at your option) any later version.


This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
GNU General Public License for more details.


You should have received a copy of the GNU General Public License
along with this program; if not, write to the Free Software
Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.
```

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

```
Gnomovision version 69, Copyright (C) year name of author
Gnomovision comes with ABSOLUTELY NO WARRANTY; for details
type `show w'. This is free software, and you are welcome
to redistribute it under certain conditions; type `show c'
for details.
```

The hypothetical commands `show w' and `show c' should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than `show w' and `show c'; they could even be mouse-clicks or menu items--whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

```
Yoyodyne, Inc., hereby disclaims all copyright
interest in the program `Gnomovision'
(which makes passes at compilers) written
by James Hacker.


signature of Ty Coon, 1 April 1989
Ty Coon, President of Vice
```

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License [http://www.fsf.org/licenses/lgpl.html] instead of this License.

GNU Free Documentation License

Version 1.2, November 2002

Copyright (C) 2000,2001,2002 Free Software Foundation, Inc. 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of “copyleft”, which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The “Document”, below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as “you”. You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A “Modified Version” of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A “Secondary Section” is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document’s overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The “Invariant Sections” are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The “Cover Texts” are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A “Transparent” copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not “Transparent” is called “Opaque”.

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The “Title Page” means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, “Title Page” means the text near the most prominent appearance of the work’s title, preceding the beginning of the body of the text.

A section “Entitled XYZ” means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as “Acknowledgements”, “Dedications”, “Endorsements”, or “History”.) To “Preserve the Title” of such a section when you modify the Document means that it remains a section “Entitled XYZ” according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document’s license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition.

Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties--for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the

same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled “History” in the various original documents, forming one section Entitled “History”; likewise combine any sections Entitled “Acknowledgements”, and any sections Entitled “Dedications”. You must delete all sections Entitled “Endorsements”.

COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an “aggregate” if the copyright resulting from the compilation is not used to limit the legal rights of the compilation’s users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document’s Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled “Acknowledgements”, “Dedications”, or “History”, the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License “or any later version” applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

```
Copyright (c) YEAR YOUR NAME.
Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License, Version 1.2
or any later version published by the Free Software Foundation;
with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.
A copy of the license is included in the section entitled “GNU
Free Documentation License”.
```

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the “with...Texts.” line with this:

```
with the Invariant Sections being LIST THEIR TITLES, with the
Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.
```

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.