Legacy Consolidations

Contents

Initial OpenIndiana versions were based on the same release process as OpenSolaris, assembled from a set of software consolidations. This complex Release Engineering has been deprecated in favour of the unified build system oi-userland: the list of consolidation and corresponding build instructions are left here as reference.

Consolidation	Description
onnv-gate	Operating System/Networking Nevada consolidation provided OpenSolars kernel and core userland components
illumos-gate	illumos-gate provides kernel and core userland components
oi-build	oi-build provided some userland components and was a oi-userland predecessor
IPS/pkg	Image Packaging System provides package management software and some zone brands definitions
SFW	Solaris Freeware consolidation provided open source software bundled with operating system
XNV	X11 Nevada consolidation provided Xorg server and related software
JDS	Java Desktop System consolidation provided Gnome2 and other desktop software
Caiman (slim_source)	slim_source provides operating system installers and Distribution Constructor
vpanels	Visual Panels provided server management GUI
sunpro/devpro	Sunpro consolidation provided Sun Studio compilers and some related libraries
xvm	Contained Sun xVM virtualization software based on Xen
g11n	Globalization consolidation contained internationalisation software

Consolidation	Description
solaris_re	Contained software related to Solaris engineering
cde	Provided Common Desktop Environment

1 Primary software consolidations

Consolidation	Original Solaris link	Original Openindiana HG link	Status	OpenIndiana Hipster link	Build instructions
onnv-gate	N/A (closed source)	N/A	Replaced by illumos-gate	N/A	N/A
illumos-gate	N/A	https://github.com/OpenIndiana/illumos-gate/	Integrated in oi-userland	https://github.com/illumos/illumos-gate/	https://illumos.org/docs/developers/build/
oi-build	https://github.com/Oracle/solaris-userland/	https://hg.openindiana.org/sustaining/oi_151a/oi-build/	Superceded by oi-userland	https://github.com/OpenIndiana/oi-userland	Building oi-build
IPS/pkg	https://github.com/Oracle/solaris-ips/	https://hg.openindiana.org/sustaining/oi_151a/pkg-gate/	Integrated in oi-userland	https://github.com/OpenIndiana/pkg5/	Building IPS/pkg
SFW	N/A (merged into solaris-userland)	https://hg.openindiana.org/sustaining/oi_151a/sfw-gate/	Merged into oi-userland	N/A	Building SFW
XNV	https://github.com/Oracle/solaris-xorg/ (merged into solaris-userland)	https://hg.openindiana.org/sustaining/oi_151a/xnv/	Merged into oi-userland	N/A	Building XNV
JDS	N/A (merged into solaris-userland)	https://hg.openindiana.org/sustaining/oi_151a/spec-files/	Merged into oi-userland	N/A	Building JDS

Consolidation	Original Solaris link	Original Openindiana HG link	Status	OpenIndiana Hipster link	Build instructions
Caiman (slim_source)	N/A	https://hg.openindiana.org/sustaining/oi_151a/slim_source/	Integrated in oi-userland	https://github.com/OpenIndiana/slim_source/	Building slim_source
vpanels	N/A (dropped)	N/A	Dropped	N/A	N/A
sunpro/devpro	N/A (closed source)	N/A	libm and make were integrated to illumos-gate, other parts are delivered in binary form (including library/medialib, system/library/c++/sunpro, developer/macro/cpp, system/library/mtsk)	N/A	N/A
xvm	N/A (dropped)	N/A	Dropped	N/A	N/A
g11n	N/A (closed source)	https://hg.openindiana.org/sustaining/oi_151a/g11n/g11n/	Parts merged into illumos-gate, parts distributed in binary forms, parts marked obsolete	N/A	N/A
solaris_re	N/A (closed source)	N/A	Relevant parts merged into oi-userland	https://github.com/OpenIndiana/oi-userland/tree/oi/hipster/components/openindiana/release	N/A
cde	N/A (closed source)	N/A	Some parts are redistributed in binary form	N/A	N/A

2 Building legacy consolidations

Note, these instructions were not tested on modern OpenIndiana versions and preserved as is just for reference. You usually don't need to build legacy consolidations them manually.

3 Building oi-build

> △ **CAUTION:**
> **oi-build is a legacy consolidation which was superceded with oi-userland !!!** For building oi-userland see Building with oi-userland

Following instructions describe building oi-build on legacy OpenIndiana /dev distribution.

oi-build is OpenIndiana's primary build framework for post-oi_151 development. It is set to replace all existing consolidations, vastly simplifying how we build the operating system.

oi-build is also tied into our continuous integration platform. When an update is committed to the oi-build mercurial repository, an automated build is kicked off. This will then automatically publish the built package to the /experimental repo, or generate an email alert if the build failed (to be completed).

3.1 oi-build overview

oi-build is a fork of Oracle's userland-gate, which we have extended and added additional software to. The layout is very similar.

Inside oi-build is a directory called `components`, under which lives a directory for each software package. Inside each of these software package directories is a main `Makefile`, as well as one or more `.p5m` IPS manifest files, and there may also be license files and patches.

The `Makefile` essentially contains a "build recipe". To build a piece of software, you simply `cd` into the directory of the software, and type `make TARGET`, where `TARGET` can be one of:

- **prep** : Download, extract and patch the software archive
- **build** : Build the software
- **install** : Install the software into the prototype directory
- **sample-manifest** : Create a sample manifest file in the build directory
- **publish** : Publish the software to the local userland IPS repo

For more details about writing Makefiles for userland, see userland Makefile targets and variables

ⓘ **NOTE:**
Before adding new packages to illumos-userland… Before considering adding a new package to oi-build, please check first whether someone else is working on the package by checking the issue tracker.

- If you don't find anyone already working on a port, please register your effort by opening an issue.
- If you wish to update an existing port, look at the log for the component Makefile ("hg log Makefile") and make sure you either contact the person who last updated the Makefile or include them on notifications for the issue by ticking their name.

This will ensure efforts aren't duplicated and help to ensure sanity and comity amongst project members.

3.2 Setting up your build environment

We strongly recommend building packages inside a fresh local zone set up exclusively for building.

Some suggestions about build environment (dataset layouts, etc.) can also be found in the illumos wiki: How to build illumos (note however that compilers required for the core OS may be different than those acceptable for userland software).

3.2.1 Ensuring your system is up to date

oi-build requires that your system be updated with the latest software from the pkg.openindiana.org/expe repository.

The procedure to do this is as follows:

```
sudo /usr/bin/pkg set-publisher -p http://pkg.openindiana.org/experimental/
sudo /usr/bin/pkg set-publisher -P oi-experimental
sudo /usr/bin/pkg set-publisher --non-sticky openindiana.org
sudo /usr/bin/pkg install -v package/pkg
sudo /usr/bin/pkg update -v
```

The first line adds the experimental repo to your package publisher list, the second line sets it as the primary publisher, and the third line allows packages from the experimental repo to replace those installed from the openindiana.org repo. The last line updates your system with the newer packages.

Note: If you are doing this in a zone and encounter an error from pkg about being unable to clone the current boot environment, you will need to update the zone from the global zone by doing:

```
ZONE=yourzonename
ZONEROOT=`zonecfg -z $ZONE info zonepath | awk '{print $NF}'`
sudo zoneadm -z $ZONE halt
sudo zoneadm -z $ZONE ready
export PYTHONPATH=${ZONEROOT}/root/usr/lib/python2.6/vendor-packages
sudo ${ZONEROOT}/root/usr/bin/pkg -R ${ZONEROOT}/root set-publisher -p
 ↪  http://pkg.openindiana.org/experimental/
```

```
sudo ${ZONEROOT}/root/usr/bin/pkg -R ${ZONEROOT}/root set-publisher -P
 ↪  oi-experimental
sudo ${ZONEROOT}/root/usr/bin/pkg -R ${ZONEROOT}/root set-publisher --non-sticky
 ↪  openindiana.org
sudo ${ZONEROOT}/root/usr/bin/pkg -R ${ZONEROOT}/root install -v package/pkg
sudo ${ZONEROOT}/root/usr/bin/pkg -R ${ZONEROOT}/root update -v
unset PYTHONPATH ZONE ZONEROOT
```

Note that the /experimental repo is under continuous development and may contain breakage. Thus we (as mentioned) strongly recommend building inside a dedicated development zone, rather than updating your main system.

3.2.2 Installing pre-requisites

First, lets install the required software list:

```
sudo /usr/bin/pkg install pkg:/archiver/gnu-tar pkg:/compress/p7zip
 ↪  pkg:/compress/unzip \
 pkg:/developer/build/ant pkg:/developer/build/autoconf
 ↪  pkg:/developer/build/automake-110 \
 pkg:/developer/build/gnu-make pkg:/developer/build/libtool
 ↪  pkg:/developer/build/make \
 pkg:/developer/gcc-3 pkg:/developer/gnome/gettext pkg:/developer/java/jdk \
 pkg:/developer/java/junit pkg:/developer/lexer/flex pkg:/developer/macro/cpp \
 pkg:/developer/macro/gnu-m4 pkg:/developer/object-file pkg:/developer/parser/bison
 ↪  \
 pkg:/developer/versioning/mercurial pkg:/file/gnu-coreutils pkg:/file/gnu-findutils
 ↪  \
 pkg:/library/libtool/libltdl pkg:/library/libxslt pkg:/library/pcre \
 pkg:/runtime/perl-512 pkg:/system/library/math/header-math pkg:/text/gawk \
 pkg:/text/gnu-diffutils pkg:/text/gnu-gettext pkg:/text/gnu-grep \
 pkg:/text/gnu-patch pkg:/text/gnu-sed pkg:/text/groff \
 pkg:/text/texinfo pkg:/library/neon pkg:/library/apr-util-13 \
 pkg:/developer/library/lint pkg:/system/header pkg:/developer/build/onbld \
 pkg:/data/docbook
```

3.2.3 Changes on Sun Studio compilers

Oracle just moved the Studio Compiler. Now you need to login to access “Oracle Solaris Studio 12.3” they can be found at

http://www.oracle.com/technetwork/server-storage/solarisstudio/downloads/index.html

The following steps are needed to get it running:

```
export DOWNLOAD_LOCATION=$HOME
cd /opt
tar xjf $DOWNLOAD_LOCATION/SolarisStudio12.3-solaris-x86-bin.tar.bz2 -C ./
ln -s /opt/SolarisStudio12.3-solaris-x86-bin/solarisstudio12.3/ ./SUNWspro
```

3.3 Downloading oi-build and preparing for first use

See the illumos wiki instructions How to build illumos for optional suggestions about build environment (dedicated ZFS dataset layouts, etc.)

Check out the repository (subdirectory oi-build must not pre-exist):

```
cd ~
hg clone https://hg.openindiana.org/oi-build
cd oi-build
```

Now we will run the setup stage, which will prepare some tools, and create an IPS pkg5 repository for first use under the i386/build directory:

```
cd $HOME/oi-build

# If you have the Sun Studio compilers installed

gmake setup

# If you don't have (access to) the Studio compilers or want to use GCC
gmake setup COMPILER=gcc
```

Important: Checking your environment

Make the check-environment target to check your environment is set up correctly:

```
gmake check-environment
```

This will report back any issues.

3.4 Adding the local repository to your publisher list

```
sudo /usr/bin/pkg set-publisher -p file://$HOME/oi-build/i386/repo
sudo /usr/bin/pkg set-publisher -P oi-build
```

You will now be able to pkg install the software you have built and published via oi-build.

3.5 Optional: Running a local pkg server for installation on other zones/hosts

If you would like to use your oi-build repository on other zones or hosts, you can run a pkg server:

```
/usr/lib/pkg.depotd -d $HOME/oi-build/i386/repo -p 10000
```

On other hosts, you can then specify http://hostname:10000 instead of the file:// address above.

See the illumos wiki instructions How to build illumos for optional suggestions about running a pkg server instance as an SMF service.

3.6 Ready to build and install your first package

Simply descend into the oi-build/components/SOFTWARE directory (where SOFTWARE is the name of the bit of software you wish to build):

```
cd $HOME/oi-build/components/SOFTWARE
gmake publish
sudo pkg install SOFTWARE
```

3.7 Contributing changes back to oi-build

Preparing you changes for review

When you think your changes are ready for review you will need to generate a diff. The most common way to do this is “hg diff”. It’s recommended that you explain the implications of your changeset if it’s not immediately obvious, and how you’ve tested it.

Committing changes

If you changeset has had either “2 thumbs up and 3 days open for review” or “4 thumbs up” then you are now ready to push you changes to the repo. Add and commit all files for your new component using:

```
cd ~/oi-build/components/SOFTWARE
hg add Makefile *.p5m <etc.>
hg commit .
```

Commit messages must follow this format:

```
<bug id> <bug summary>
Reviewed by: Reviewer Name <reviewer.name@fake.net>
Reviewed by: Reviewer Name <reviewer.name@fake.net>
Reviewed by: Reviewer Name <reviewer.name@fake.net>
Approved by: My Name <my.name@mydomain.net>
```

When committing changes on behalf of someone else the commit message remains the same but you must pass the committer’s name to hg commit. hg commit -u ‘Contributor Name contributor.name@fake.net’

4 Building IPS/pkg

> ⓘ **NOTE:**
> For building IPS delivered with modern OpenIndiana, use openindiana/pkg oi-userland component.

Following instructions describe building pkg on legacy OpenIndiana /dev distribution.

Make sure you have Sun Studio installed.

Install dependencies:

```
sudo pkg install                \
developer/build/autoconf          \
```

```
developer/build/automake-110        \
developer/opensolaris/pkg5          \
developer/swig                      \
developer/versioning/mercurial      \
gnome/accessibility/gnome-a11y-libs \
library/python-2/python-gnome-extras-26 \
library/python-2/python-notify-26   \
package/pkg/package-manager         \
package/pkg/update-manager          \
developer/python/pylint             \
service/network/dns/mdns            \
system/library/math/header-math     \
text/gnu-patch \
developer/gnome/gnome-doc-utils \
system/zones/internal
```

Set up your environment:

```
GATE=pkg
BUILDNUM=151

pfexec /sbin/zfs create -o atime=off -p rpool/export/builds/$GATE
pfexec chown `id -u`:`id -g` /export/builds/$GATE
cd /export/builds/$GATE

unset CC
unset CXX
PATH=/usr/bin:/usr/sbin:/opt/SUNWspro/bin:/usr/ccs/bin
export PATH
```

Get the source and update it to the right branch:

```
hg clone https://hg.openindiana.org/sustaining/oi_151a/pkg-gate/ pkg-gate
cd pkg-gate
hg update oi_151a
```

Start the building process:

```
cd src
dmake install
make -e packages BUILDNUM=${BUILDNUM}
export PATH=`pwd`/../proto/root_`uname -p`/usr/bin:$PATH
export PYTHONPATH=`pwd`/../proto/root_`uname
 ↪  -p`/usr/lib/python2.6/vendor-packages:$PYTHONPATH
cd pkg
make BUILDNUM=${BUILDNUM}
make repository-metadata
```

Optional unit tests, which take forever:

```
cd ..
make test-verbose
```

5 Building SFW

Following instructions describe building SFW on legacy OpenIndiana /dev distribution.

SFW is the Solars Freeware consolidation, which includes a lot of FOSS software. Misc note: By default on re-invoking env -i /opt/onbld/bin/nightly yourenvscrip.sh the build process will clear up the previous build run by removing directories with rm -rf

Building SFW is a lot like building ONNV. It's recommended you build SFW on a version of ONNV that's not too far behind it (e.g. build SFW 145 on ONNV 144). Please ensure Sun Studio is set up and pkg has been installed from pkg-gate, as per the ONNV build instructions. Also ensure an up to date onbld is present for the version of SFW you're building (if you've built ONNV on the same box then that will be the case).

5.1 Installing dependencies

You'll need to ensure the following dependencies are installed:

```
sudo pkg install \
archiver/gnu-tar \
compatibility/ucb \
compress/unzip \
data/docbook \
database/mysql-51/library \
database/postgres-82/developer \
developer/build/ant \
developer/build/autoconf \
developer/build/automake-19 \
developer/build/automake-110 \
developer/build/cmake \
developer/build/gnu-make \
developer/build/libtool \
developer/build/onbld \
developer/gcc-3 \
developer/gnome/gettext \
developer/java/junit \
developer/lexer/flex \
developer/object-file \
developer/parser/bison \
developer/versioning/subversion \
file/gnu-coreutils \
file/gnu-findutils \
library/c++/sigcpp \
library/gd \
library/glib1 \
library/graphics/wxwidgets \
library/guile \
library/libevent \
library/libtorrent \
library/motif \
```

```
library/mozilla-nss/header-nss \
library/nspr \
library/nspr/header-nspr \
library/pcre \
library/perl-5/database \
library/python-2/libxml2-26 \
library/python-2/setuptools-26 \
library/slang \
library/tooltalk \
print/cups \
print/filter/ghostscript \
runtime/erlang \
runtime/lua \
runtime/ocaml \
runtime/ocaml/lablgtk \
service/network/slp \
system/header/header-audio \
system/header/header-ugen \
system/header/header-usb \
system/header/header-agp \
system/library/libpcap \
system/library/math/header-math \
system/library/usb/libusb \
system/network/avahi \
text/gawk \
text/gnu-diffutils \
text/gnu-gettext \
text/gnu-grep \
text/gnu-patch \
text/gnu-sed \
text/groff \
text/texinfo \
web/java-servlet/tomcat \
web/server/apache-13 \
web/server/apache-22 \
x11/library/mesa \
x11/optional-clients
```

Then run the following to set up the build environment:

```
pfexec sh /usr/share/sgml/docbook/docbook-catalog-install.sh
```

Note: This command seems to be unnecessary.

Note: If repeating builds do not repeat the docbook-catalog-install.sh, it will break stuff!

You will likely find that the build never completes. Instead, it hangs with a spinning ‘yes’ command. This happens because php-5.2.17 must be built with version 2.13 of the autoconf tools. Version 2.69 is installed by the package listed above. You will need to build 2.1.3 from source and install it where it won’t interfere with 2.69. /opt/auto is a suitable place. Then you will need to patch usr/src/cmd/php5/Makefile.sfw so that the php build will use 2.13. This patch file is suitable:

```
--- Makefile.sfw-orig      ::
+++ Makefile.sfw           ::
@@ -18,9 +18,9 @@
 #
 # CDDL HEADER END
 #
+# Copyright 2016 Gary Mills
 # Copyright (c) 2007, 2011, Oracle and/or its affiliates. All rights reserved.
 #
-#pragma ident  "@(#)Makefile.sfw       1.47    11/06/15 SMI"

 PHP_REL=5.2
 #
@@ -118,7 +118,9 @@

 PRECONF_ENVLINE= \
        PATH=/usr/gnu/bin:/usr/bin \
-       MAKE=$(GMAKE)
+       MAKE=$(GMAKE) \
+       PHP_AUTOCONF=/opt/auto/bin/autoconf \
+       PHP_AUTOHEADER=/opt/auto/bin/autoheader

 CONF_ENVLINE= \
        CC=$(CC) \
```

5.2 Doing a build:

You can obtain the sfwnv source from either the Sun website (selecting the correct directory and files for the version you wish to build) or use one of the hg repos like:

```
hg clone https://hg.openindiana.org/sustaining/oi_151a/sfw-gate/
```

cd to the directory you either unpacked or cloned then edit or make a copy of usr/src/tools/env/sfw-opensolaris.sh and invoke the build with

```
env -i /opt/onbld/bin/nightly ./sfw-opensolaris.sh &
```

5.3 Variables

In `usr/src/tools/env/yourenvscrip.sh` `MAKEFLAGS=k` means continue on error `DMAKE_MAX_JOBS` is worth cranking if you're on a multicore system See scripts for other options available as it's well documented.

6 Building XNV

Following instructions describe building XNV on legacy OpenIndiana /dev distribution.

6.1 Install pre-requisites

Note: you must build this XNV release with Sun Studio 12 (Sept 2009 edition with CBE patches), and NOT Sun Studio 12.1. If you build it with 12.1 you'll suffer frequent freezes.

The following should install all the dependencies required to build X, with the exception of Sun Studio 12 which you will have to obtain separately.

```
pfexec pkg install developer/opensolaris/X developer/build/cbe x11-network-proxies
```

6.2 Do the build

```
cd ~
hg clone https://hg.openindiana.org/sustaining/oi_151a/xnv
cd xnv
hg update oi_151a
make setup X_BUILD_OPTIONS=openindiana
```

Fetch the tar balls by executing:

```
./download-tarballs
```

Start the XNV build process:

```
export PATH=/opt/SUNWspro/bin:/usr/bin:/usr/sbin:/usr/ccs/bin:bin:/usr/dt/bin/:/us
↪  r/openwin/bin:/opt/onbld/bin:/opt/onbld/bin/i386:/usr/sfw/bin
./buildit
```

As xnv uses a parallel build, if the build process fails, run

```
open-src/util/build-tools/find-build-errors
```

to find errors.

Create the IPS repository:

```
cd pkg
make
```

7 Building JDS

Following instructions describe building JDS on legacy OpenIndiana /dev distribution.

Steps for building JDS

- Create a fresh zone (do not use the global zone, during the build files will be installed directly into the running system bypassing the packaging system!).
- Obtain oi-cbe from https://hg.openindiana.org/oi-cbe/
- Download sunstudio12u1-patched-ii-2010Feb-sol-x86.tar.gz and desktop-cbe-mini-1.8.0.tar.bz2 and place them inside oi-cbe/sources

Copy oi-cbe into the zone and execute setup-buildenv.sh jds in order to set up the build environment.

```
# buildzone=jdszone       # name of the designated build zone
# zonepath=/path/to/zone  # path to the designated build zone
# cp -r oi-cbe ${zonepath}/root/var/tmp/
# zlogin $buildzone 'cd /var/tmp/oi-cbe && ./setup-buildenv.sh jds'
```

Download and place all JDS source files into ${HOME}/packages/SOURCES

```
# zlogin -l abuild $buildzone 'cd ${HOME}/packages/SOURCES && wget -r -l1 -np -nd
 ↪  http://dlc.openindiana.org/oi/jds/downloads/sources/'
```

Check out the JDS spec files and the OpenIndiana patchset.

```
# zlogin -l abuild $buildzone 'hg clone
 ↪  https://hg.openindiana.org/spec-files-mozilla-l10n/
 ↪  ${HOME}/spec-files-mozilla-l10n'
# zlogin -l abuild $buildzone 'hg clone https://hg.openindiana.org/spec-files/
 ↪  ${HOME}/spec-files'
# zlogin -l abuild $buildzone 'hg clone https://hg.openindiana.org/mq_spec-files/
 ↪  ${HOME}/spec-files/.hg/patches'
```

```
Update to the desired version and apply all patches.
```

```
# zlogin -l abuild $buildzone 'cd ${HOME}/spec-files && hg -R .hg/patches up oi_148
 ↪  && hg up NEVADA_148 && hg --config extensions.mq=1 --config diff.git=1 qpush -a'
```

Prepare and build JDS.

```
# zlogin -l abuild $buildzone '. /opt/dtbld/bin/env.sh; cd ${HOME}/spec-files &&
 ↪  make'
# zlogin -l abuild $buildzone '. /opt/dtbld/bin/env.sh; rm ${HOME}/status.html; cd
 ↪  ${HOME}/spec-files && pkgtool -v --topdir=${HOME}/packages --tarballdirs
 ↪  ${HOME}/packages/SOURCES:${PWD}/manpages/sun-manpage-tarballs:${PWD}/manpages-
 ↪  roff/sun-manpage-tarballs:${PWD}/po-sun/po-sun-tarballs:${tarballdirs} --update
 ↪  --rmlog --logdir=${HOME}/logs --logdir-url=/logs --live
 ↪  --summary-log=${HOME}/status.html --nopkg --with-l10n
 ↪  --with-openindiana-branding build specs/*.spec'
# zlogin -l abuild $buildzone '. /opt/dtbld/bin/env.sh; rm ${HOME}/status.html; cd
 ↪  ${HOME}/spec-files-mozilla-l10n && pkgtool -v --topdir=${HOME}/packages
 ↪  --tarballdirs ${HOME}/packages/SOURCES:${tarballdirs} --update --rmlog
 ↪  --logdir=${HOME}/logs --logdir_url=/logs --live
 ↪  --summary-log=${HOME}/status.html --nopkg --with-l10n build specs/*.spec'
```

SVR4 packages will be placed into ${HOME}/packages.

8 Building slim_source

> ⓘ **NOTE:**
> For building slim_source used in modern OpenIndiana, use openindiana/slim_source oi-userland component.

Following instructions describe building slim_source on legacy OpenIndiana /dev distribution.

Build Environment

We would always recommend building in a Zone. To build slim_source as per the instructions below, please make sure you are running oi_151. You will also need to make sure you have Sun Studio installed.

Get slim_source source:

```
export BUILD_ID=oi_151a
export SNV=151

hg clone https://hg.openindiana.org/sustaining/oi_151a/slim_source/
cd slim_source
hg update $BUILD_ID
```

Install required IPS packages:

```
sudo pkg install archiver/gnu-tar \
developer/build/onbld \
developer/object-file \
developer/swig \
developer/versioning/mercurial \
gnome/config/gconf \
install/beadm \
library/desktop/gtk2 \
library/desktop/libglade \
library/glib2 \
library/gnome/gnome-libs \
package/svr4 \
service/network/smtp/sendmail \
system/boot/wanboot/internal \
system/library/install \
system/library/install/libinstzones \
system/library/libdiskmgt/header-libdiskmgt \
system/library/storage/ima/header-ima \
system/zones/internal \
text/gnu-gettext \
text/gnu-grep \
text/gnu-sed
```

Prepare the build script:

```
cd usr/src
cp tools/env/developer.sh .
gsed -i 's%export CODEMGR_WS=.*%export CODEMGR_WS="`hg root`"%g' developer.sh
echo 'export CW_NO_SHADOW=1' >>developer.sh
echo "export INSTALL_BUILDNUM=$SNV" >>developer.sh
echo 'export NIGHTLY_OPTIONS="-ANndlmp +t"' >> developer.sh
echo 'export SPRO_ROOT=/opt/sunstudio12.1' >> developer.sh
```

Start the build:

```
/opt/onbld/bin/nightly developer.sh
```

Once the build is complete, you can check the logs in ../../log/*

If the build was successful, the end result are two repos: under the base slim_source directory: `packages/i386/nightly-nd/repo.extra` and `packages/i386/nightly-nd/repo.redist`