# Build-FAQ

Mask the recipe in platform specific MACHINE conf file and run the source command again to build.

For example, to MASK ccsp-gwprovapp-ethwan in Rpi build, add below line in Rpi specific Machine conf file. Add below line in file

“**meta-cmf-raspberrypi/conf/machine/raspberrypi-rdk-broadband.conf ”**

**BBMASK .= "|.meta-rdk-broadband/recipes-ccsp/ccsp/ccsp-gwprovapp-ethwan.bb"**

Run the source command again to build

**source meta-cmf-raspberrypi/setup-environment -> ** select respective conf file

If you encounter below error,

repo: warning: Python 2 is no longer supported; Please upgrade to Python 3.6+.
Downloading Repo source from https://gerrit.googlesource.com/git-repo
remote: Counting objects: 1, done
remote: Finding sources: 100% (41/41)
remote: Total 41 (delta 14), reused 41 (delta 14)
Unpacking objects: 100% (41/41), done.
Traceback (most recent call last):
File "<File Path>", line 56, in <module>
from subcmds.version import Version
File "<File Path>", line 38, in <module>
['%s' % name])
File "<File Path>", line 27, in <module>
from hooks import RepoHook
File "<File Path>", line 472
file=sys.stderr)
^
SyntaxError: invalid syntax

`Upgrade the repo using below command to use with python3.`

```
curl https://storage.googleapis.com/git-repo-downloads/repo-1 > ~/bin/repochmod a+x ~/bin/repopython3 ~/bin
/repo init -u <repo URL>python3 ~/bin/repo sync
```

Make sure that the build machine is having required packages installed for specific Yocto flavor.
Eg: https://docs.yoctoproject.org/3.2.3/ref-manual/ref-system-requirements.html

Metadata represents the versions of the various components in a distribution, such as the particular versions of the Linux kernel or libraries. The project supplies an example set of metadata that can generate several example distributions. The actual metadata used for the construction of a custom distribution may be supplied by a commercial vendor or created by an embedded developer. The root filesystem is defined in the metadata for a given build of a distribution.
There are tools within BitBake that enable this level of details.

- “bitbake -g targetname” creates depends.dot and task-depends.dot files in the current directory. These files show which packages and tasks depend on which other packages and tasks and are useful for debugging purposes.
- "bitbake -g -u depexp targetname" shows results in a more human-readable, GUI style. A simple mount of the resulting root image will show how much storage space is being used.

In addition, the toaster is a new graphical user interface for BitBake that makes these tools much easier to use.

As with any complex system, the real answer is *it depends*, but of course that is not very helpful. The simplest method for adding a single package to your build is to add

the below line to `conf/local.conf`:

- IMAGE_INSTALL_append = " package"

Use your own package name in place of **package**. Note the leading space before the package name. If you want to add multiple packages, you can use multiple lines like the above, or list all packages on a single line with:

- IMAGE_INSTALL_append = " package1 package2 package3"

Although If you add in local.conf , that is not permanent change. For permanent addition of that package in final rootfs, you need to be added in image recipe or any package group which is included in the image recipe.
Prebuilds are handled internally by Yocto by using sstate-cache. If a prebuilt from a known good build is available, the build can point to that folder via the conf file inside the *./build<buildtype>/conf/* folder so that the prebuilts are picked up from the location

This depend entirely on multiple factors like capacity of build machine, first time build or repeated build in the same work space as well as changes in components on which the component in question depends on( if there is a change, the depending component is first built and then the dependent component ) and hence cannot be answered directly.
When you checkout sandbox every component is independently buildable, and bitbake ( OE's build engine) is responsible to identify and sort the component dependency chain and ensure its built along. if you were to build a single component the commands are

```
bitbake <component>
```

where component is in one to one relation with .bb ( recipe ) file that can be found in the Yocto/OE metadata ( meta-rdk* layers ) e.g. if you were to build rdkbrowser then you would see that its recipe is housed in generic layer called meta-rdk-cmf and recipe is called rdkbrowser.bb so the command would be

```
bitbake rdkbrowser
```

However this will only generate CMF component and for packaging it up into final image you still have to build the image component to repackage rdkbrowser

```
bitbake rdk-generic-hybrid-wpe-image
```

would generate the CMF generic image for hybrid devices. it will only rebuild the affected components when building the image if nothing has changed it will not recreate the image.
bitbake has division of work into individual tasks for a component. Secondly, the recipes are wired to notice changes in upstream repository as well when you do repo sync. You can use below command to see what all individual tasks are available.

```
bitbake -c listtasks <component>
```

It will show an output like

```
do_build                       Default task for a recipe - depends on all other normal tasks required to 'build'
a recipe

do_bundle_initramfs            Combines an initial ramdisk image and kernel together to form a single image

do_checkuri                    Validates the SRC_URI value

do_checkuriall                 Validates the SRC_URI value for all recipes required to build a target

do_clean                       Removes all output files for a target

do_cleanall                    Removes all output files, shared state cache, and downloaded source files for a
target

do_cleansstate                 Removes all output files and shared state cache for a target

do_compile                     Compiles the source in the compilation directory
```

You can rerun any of the above tasks

```
bitbake -C compile rdkbrowser
```

would force recompile of servicemanager, if you wish to perform all the build steps for a component you can do that too by

```
bitbake -c cleansstate rdkbrowser; bitbake rdkbrowser
```

Similarly, in general you can have:

bitbake <target> -c<task_to_be_executed>

This will ensure do_<task_to_be_executed>() will be called.

task_to_be_executed can  fetch, unpack, configure, compile, install, package etc

**If we were to draw parallels**

--rebuild = bitbake -c cleansstate <component> ; bitbake <component>

--build-only = bitbake <component>

--skip-package is delegated to shared state mechanism to figure out at present.
Unless you do 'repo sync' sources will not be updated

'repo sync' can cover only components with external src support, it means that in cases when SRCREV is set to AUTOREV and component doesn't support external src, then bitbake will try to update sources from a remote repository during build time.

There is no documentation for AUTOREV, but it's doing only one function - check remote repository for any new sources updates.

```
AUTOREV = "${@bb.fetch2.get_autorev(d)}"
```

You can prevent this behaviour by changing BB_SRCREV_POLICY variable in you local <sandbox>/conf/local.conf

```
BB_SRCREV_POLICY = "cache"
```

```
BB_SRCREV_POLICY
Defines the behavior of the fetcher when it interacts with source control systems and dynamic source revisions.
The BB_SRCREV_POLICY variable is useful when working without a network.
The variable can be set using one of two policies:


cache - Retains the value the system obtained previously rather than querying the source control system each
time.

clear - Queries the source controls system every time. With this policy, there is no cache. The "clear" policy
is the default.
```

These are OE metadata variables, bitbake has preprocessing options where it expands all the local bitbake variables so you could take advantage of that option to figure it out

```
bitbake -e rf4ce | grep "^S ="
```

```
bitbake -e rf4ce | grep "^WORKDIR ="
```

It can be used to check for any bitbake variable, alternatively you can pipe the whole bitbake -e output to a file and inspect the file in your favourite editor

- You can find log files for each task in the recipe's `temp` directory. Log files are named `log.taskname`*-Eg:compile logs are present in the file log.do_compile.* Bitbake maintains logs separately for each of the tasks that run while building the component. These tasks are typically the fetch task, the compile task, install task and so on.
- Eg: For a RPI device, typically logs are present under the build-raspberrypi-rdk-hybrid//tmp/work/<*-rdk-linux> and under the component directory. Other devices would have logs present under similar folders. For instance, logs are present under build-raspberrypi-rdk-hybrid/tmp/work/<*-rdk-linux>/rdkbrowser/1.99+gitAUTOINC+8c3f17fdc6_fa6c8b4334_dc33f7d6bc_4bc1f2f4c9-r0/temp

The warnings like -Werror -Wall -Wextra are turned on for compiler for most of the Ccsp components .All these warnings must be fixed for successful compilation as all warnings are treated as errors.

Werrors due to unused parameters in code can be fixed by using:

UNREFERENCED_PARAMETER(user_data); ,if user_data variable is not used in the scope of definition of the function .

```
* satisfy_dependencies_for: Cannot satisfy the following dependencies for packagegroup-...:

*             net-snmp *

* opkg_install_cmd: Cannot install package packagegroup-....
```

**The above error indicates that :**

You asked for adding net-snmp ( the package ) not ( the recipe ) now net-snmp ( the recipe ) may generate a number of ( packages ) so you should add the packages ( runtime items) to the package groups and not the recipes ( build time items). Usually yocto/OE does generate a output package with same name as input recipe so for net-snmp.bb there will be a net-snmp ipk but thats just a common case not a hard and fast rule.

Now in this particular case when a package has nothing to emit into the ${PN} package the package is left empty and hence not emitted. If you want to emit the package regardless you have to add

```
ALLOW_EMPTY_<package> = "1" in the recipe,but this is less of a usecase to demand empty packages. If you
expressed the packagegroup RDEPENDS correctly you would not need it.
```

sstate-cache is always adding new versions and hence is growing in size always, thankfully we have a tool in Yocto to manage it, here is a sample on how to do it for **raspberrypi** you can set WORKSPACE and MACH variables to point to the values for machine
**sstate-manage.sh**

```
# cd into top of workspace say ${WORKSPACE}
MACH=raspberrypi
MACHINE=${MACH} . ./meta-rdk/setup-environment
EXTRALAYERS=`bitbake -e | grep '^BBLAYERS='| tr -s ' ' ','| tr -d '"'| sed -e 's%^BBLAYERS=%%' -e 's%,$%%'`
# remove stale sstate on mirror
${WORKSPACE}/openembedded-core/scripts/sstate-cache-management.sh -v --yes \
          --extra-layer=${EXTRALAYERS} \
          --cache-dir=${WORKSPACE}/sstate-cache \
          --stamps-dir=${WORKSPACE}/build-${MACH}/tmp/stamps
```

To enable kernel/busybox features you can append metadata to the recipes (i.e. .bb files) by simply creating an append file (i.e. .bbappend files) and including metadata in it. If the features needs to be enabled across all the platforms then add in **meta-rdk-rpi** meta or if it's specific to a platform then append to the recipes available in OEM layer specific to the platform.

Below example shows how to enable IPSec on Yocto builds:

- Create a stblinux_%.bbappend
- Create a file called enable_netkey.cfg with following content
  CONFIG_NET_KEY=y
- Add SRC_URI_append = "file://enable_netkey.cfg "

Make sure .bbappend has the same root name as their corresponding .bb recipes.

- Sometimes when you have a working branch which is not checked out or has uncommitted changes then repo will fail when you try to sync to the latest code base.

- Sample failure logs:

    error: generic/devicesettings/generic/: contains uncommitted changes

    error: generic/rdkbrowser/: branch master is published (but not merged) and is now 11 commits behind

    error: meta-rdk-oem-X/: contains uncommitted changes

- To resolve this you need to checkout the branch and rebase it to the master using below commands.

    git rebase —abort

    git rebase rdkgerrit/master ( or rdkgerrit/stable2)

- During the process git might throw conflict errors if it cannot merge files automatically. Then you need to merge manually using Vim or any other text editors. But it can be simple if you know exactly what changes needs to be saved / removed.
- For example you can use below command to keep your changes

    git checkout --ours FILE

- If you want to run on multiple files then use below command.

    grep -lr '<<<<<<<<'.  |  xargs git checkout --ours

- Similarly,  you can use below commands if you want to keep other changes.

    git checkout --yours FILE

    grep -lr '<<<<<<<<'.  |  xargs git checkout –theirs

- Verify the files from which recipe they are being generated under build-<platform>/buildhistory/packages/mips32el-rdk-linux folder
- Add the below line format under that particular recipe.

    FILES_${PN}-dev += " file1 file2 etc"

- Verify that *cat <package>-dev/latest*  should contain these files under FILELIST.

Yocto project build system has a utility which can provide information about which package (ipk or rpm) is providing a given file, this helps in finding further information on packaging e.g. if you want to do more finer packaging, run the following command in your build environment

    oe-pkgdata-util find-path /lib/libc.so.6
    glibc: /lib/libc.so.6

Common build failures are reported in Yocto builds. Some build failures are hard to analyze with logs, unless we get access to the failure workspace. In most cases they are hard to reproduce on local workspace. We go through multiple iteration of builds, lock down the node and then debug. To debug these failures use Packages file found under *tmp/deploy/ipk* directory on you local workspace .
The signal core dump that are generated under /tmp can be decoded using gdb
Procedure :

1. In <image>.bbappend file
   IMAGE_INSTALL_append = " gdb"
2. In local.conf
   INCOMPATIBLE_LICENSE = "GPL-3.0 LGPL-3.0 AGPL-3.0"
   INCOMPATIBLE_LICENSE_pn-gdb = ""
   EXTRA_IMAGE_FEATURES += "tools-debug"
   EXTRA_IMAGE_FEATURES += "dbg-pkgs"
3. In <component>.bbappend
   CFLAGS += " -D_DEBUG -g "
4. Compile and flash the binary to device
5. Run gdb -c <path to signal dump> <binary>

This should be because of architecture bit mismatch . To overcome this should either choose the right target platform or put the executable file as a tar file in bb file.
**Issue:**

While building the stack, the bitbake process will be aborted if the disk space runs low beyond the minimum requirement.

**Example console log:**

WARNING: The free space of /mnt/home/gpsahu01/cmf/test/build-qemux86hyb/tmp (/dev/vdb) is running low (0.999GB left)

ERROR: No new tasks can be executed since the disk space monitor action is "STOPTASKS"!

WARNING: The free space of /mnt/home/gpsahu01/cmf/test/downloads (/dev/vdb) is running low (0.095GB left)

ERROR: Immediately abort since the disk space monitor action is "ABORT"!

NOTE: Sending SIGTERM to remaining 1 tasks

**Possible solution:**

Yocto stack requires more than 30 GB of free disk space for build to complete, so it is required to keep sufficient disk space available before starting the build process.

**Issue:**

Bitbake complains about a missing “sys/cdefs.h” and the error can be encountered in random recipes when we move from one build host to other.

**Example console log:**

```
compilation terminated.
| In file included from /usr/include/stdio.h:27:0,
|                  from ./src/kern_head.c:13:
| /usr/include/features.h:374:25: fatal error: sys/cdefs.h: No such file or directory
|  #  include <sys/cdefs.h>
|                          ^
| compilation terminated.
| In file included from /usr/include/stdio.h:27:0,
|                  from ./src/sstrip.c:9:
| /usr/include/features.h:374:25: fatal error: sys/cdefs.h: No such file or directory
|  #  include <sys/cdefs.h>
|                          ^
| compilation terminated.
| make: *** [encode] Error 1
| make: *** Waiting for unfinished jobs....
| make: *** [kern_head] Error 1
| make: *** [sstrip] Error 1
| ERROR: oe_runmake failed
| WARNING: exit code 1 from a shell command
```

**Possible Solution:**

This issue may be caused by a missing “g++-multilib” package in the build host (observed in Ubuntu 14.4). Installing the package with “sudo apt-get install g++-multilib” should resolve this issue. Also the build machines should be configured following the procedure as per Setup guide to avoid similar issues.

**Issue:**

The bitbake process terminates after complaining about a non-existent path or environment variable.

**Example console log #1:**

ERROR: Function failed: iarmbus: LIC_FILES_CHKSUM points to an invalid file: ${RDK_ROOT_PATH}/components/generic/iarmbus/core/include/libIARM.h

**Example console log #2:**

ERROR: ParseError at /mnt/home/gpsahu01/cmf/test/meta-virtualization/recipes-extended/images/cloud-image-controller.bb:29: Could not inherit file classes/image-vm.bbclass

**Possible solution:**

This may be due to a wrongly setup environment e.g. we have executed “meta-rdk/setup-environment” instead of sourcing “meta-cmf/setup-environment”

**Issue:**

The issue is observed during setup and machine selection stage, setup-environment script will throw unexpected error about non-existing layer paths.

**Example console log:**

```
gpsahu01@dvm-wcdcc-tata-001:~/cmf/emulator-2.1-20160919$ source meta-cmf/setup-environment

1) meta-raspberrypi/conf/machine/raspberrypi0.conf          7) meta-rdk-bsp-emulator/conf/machine/qemux86hyb.conf

2) meta-raspberrypi/conf/machine/raspberrypi2.conf          8) meta-rdk-bsp-emulator/conf/machine/qemux86mc.conf

[…]

Please enter your choice of machine [1..11]: 7

### Shell environment set up for builds. ###

Writing auto.conf ...

Writing versions.txt ...

-bash: cd: ../meta-browser//: No such file or directory

fatal: Not a git repository (or any parent up to mount point /mnt)

Stopping at filesystem boundary (GIT_DISCOVERY_ACROSS_FILESYSTEM not set).

-bash: ../patches/rdk-oe/meta-linaro//*.patch: No such file or directory
```

-bash: cd: ../meta-openembedded//: No such file or directory

fatal: Not a git repository (or any parent up to mount point /mnt)

Stopping at filesystem boundary (GIT_DISCOVERY_ACROSS_FILESYSTEM not set).

**Possible solution:**

If the host PC has set colored terminal output for commands then it may cause unexpected errors being shown during execution of meta-cmf/setup-environment script. To fix the problem we can run following command:

**gpsahu01@dvm-wcdcc-tata-001:~/cmf/emulator-2.1-20160919$ alias ls='ls --color=no'**
**Issue:**

Fetch timeout or failure can happen due to following network problems.

- Network not accessible
- Restriction in Firewall
- Invalid Proxy configuration
- Unable to resolve DNS in IPv6 networks

**Example console log:**

Fetching projects: 96% (91/94) Fetching project openembedded/meta-linaro
Fetching projects: 97% (92/94) fatal: read error: Connection timed out
fatal: read error: Connection timed out
fatal: read error: Connection timed out
error: Cannot fetch meta-virtualization
warn: --force-broken, continuing to sync
Fetching projects: 98% (93/94) error: Cannot fetch meta-java
warn: --force-broken, continuing to sync
Fetching projects: 100% (94/94)
error: Exited sync due to fetch errors

**Possible solution:**

- Using VPN may have some restrictions sometime it may not allow GIT access.
- Ensure that the ports for HTTPS, SSH, HTTP are opened by the firewall and the policy doesn't block common open source repositories.
- In case of IPv6 networks issues, force GIT to use IPv4.

Also Following options can be considered while debugging:

**Option #1)** Need to flush the IP rules:
enter the command
$ iptables -F
and check
$) git clone git://git.lighttpd.net/lighttpd/lighttpd-1.x.git

**Option #2)** Check for the port 22 is open or not by doing nmap
$ nmap -p 22 10.11.107.0-255"
(check for ipaddress 10.11.106.62)
$ ssh -v git@github.com
if it adds it will ask for input()yes/no- Type yes
$ git clone git://github.com/lighttpd/lighttpd1.4.git;branch=lighttpd-1.5.x

**Option #3)** replacing the git// with https:// which uses port 443
$ git config --global url."https://".instead of git://
and try $ "git clone git://git.lighttpd.net/lighttpd/lighttpd-1.x.git"

**Option #4)** in a few cases, the access to GIT repository is via SSH. To use SSH URLs with GIT repository, an SSH key-pair must be generated on the build PC and add the public key to your GitHub account.
For information on setting up an SSH key-pair, see "Generating an SSH key."
https://help.github.com/articles/generating-an-ssh-key/
**Issue:**

When we try to build a very old branch of the code, the manifest file will not be up-to-date as few of the open-source URLs might not be continuing support of older branches or versions of software.

**Example console log:**

WARNING: Failed to fetch URL git://code.qt.io/qt/qtlocation.git;branch=stable, attempting MIRRORS if available
ERROR: Fetcher failure: Unable to find revision f28408346243cf090326f4738fd838219c21e00f in branch stable even from upstream
ERROR: Function failed: Fetcher failure for URL: 'git://code.qt.io/qt/qtlocation.git;branch=stable'. Unable to fetch URL from any source

**Possible solution:**

It is recommended to build with more recent branches, as the code will be well maintained and will have updated features.
**Issue:**

An Open source URL is broken either due to the website is down temporarily or it is permanently removed.

**Example console log:**

WARNING: Failed to fetch URL http://directfb.org/downloads/Core/linux-fusion/linux-fusion-9.0.3.tar.gz, attempting MIRRORS if available
ERROR: Fetcher failure: Fetch command failed with exit code 4, no output
ERROR: Function failed: Fetcher failure for URL: 'http://directfb.org/downloads/Core/linux-fusion/linux-fusion-9.0.3.tar.gz'. Unable to fetch URL from any source

**Possible Solution:**

**Temporary workaround:** In case of archives (.tar or .zip etc.), if the file is available from a previously built stack then it can be copied and an empty file with the name <archive-name>.done has to be created to bypass looking for downloading the file.

**Fixing the recipe:** If the problematic recipe is available from any other alternative mirror, update the same in SRC_URI part of the recipe. Few components may be available in common mirrors such as github, web.archive.org, oipf.tv etc.
Looks like memory issue hence changing Ubuntu to 64 bit version should resolve the issue. The below are the Ubuntu configurations,

- Ubuntu 16.04 - 64 bit
- 6GB RAM

16GB Swap
To include a specific feature that is not available in base build, enable the feature specific DISTRO flag in platform specific config file. For example to include USPA feature in Rpi build,

Add the DISTRO specific flag in Rpi platform specific conf file

In File **meta-cmf-raspberrypi/conf/distro/include/rdk-rpi.inc**

Add **DISTRO_FEATURES_append = " usppa" (to include the feature if not there)**

**DISTRO_FEATURES_remove = " usppa" (to remove the feature)**

**Make sure the recipe is part of the package build**

In File **meta-rdk/recipes-core/packagegroups/packagegroup-rdk-ccsp-broadband.bb** must be included as a DISTRO protected feature **RDEPENDS_packagegroup-rdk-ccsp-broadband += " \ ${@bb.utils.contains('DISTRO_FEATURES', 'usppa', "usp-pa", "", d)}"**

```
ERROR: ParseError at /home/a1602446/build-raspberrypi-rdk-broadband/conf/local.conf:247: Could not include required file conf/distro/include/##RDK_FLAVOR##.inc
```

Above error occurs intermittently, which can be fixed by retrying the source command for setup_environment file.

In case of Rpi, it is

- **source meta-cmf-raspberrypi/setup-environment**

Please refer Compile-time Build Variants Flags
**Issue:**

Bitbake fails on populate sysroot state when building with an un-clean stack.

**Example console log:**

ERROR: Function failed: llvm_sysroot_preprocess (log file is located at /mnt/home/gpsahu01/cmf/test/build-qemux86hyb/tmp/work/i586-rdk-linux/llvm3.3/3.3-r0/temp/log.do_populate_sysroot.9648)

**Possible solution:**

This may happen when a previous build process was unexpectedly terminated or aborted. Re-build after cleaning the problematic recipe or image (bitbake <recipe> -c cleanall) would fix the issue.
**Issue:**

Bitbake terminates the compilation process on ‘do_patch’ task. This may happen in following cases:

- When using an old recipe file where the SRC_URI link has updated its folder structure.

- Wrongly formatted patch file (run dos2unix for conversion)

- Incorrect patch level (p0, p1, etc.)

**Example console log:**

ERROR: Command Error: exit status: 1 Output:
Applying patch 0001-src-Makefile.am-Dont-check-if-we-are-cross-compiling.patch
can't find file to patch at input line 18
Perhaps you used the wrong -p or --strip option?
The text leading up to this was:
--------------------------

| |
|---|
| diff --git a/src/Makefile.am b/src/Makefile.am |
| index eb50e37..c1e3d64 100644 |
| — a/src/Makefile.am |
| +++ b/src/Makefile.am<br>--------------------------<br>No file to patch. Skipping patch.<br>2 out of 2 hunks ignored<br>Patch 0001-src-Makefile.am-Dont-check-if-we-are-cross-compiling.patch does not apply (enforce with -f)<br>ERROR: Function failed: patch_do_patch |

**Possible Solution:**

From the above output it seems that the file to which patch will be applied is not found, possible reason may be the source folder structure doesn't match with the destination folder structure. E.g. the source directory or **${S}** starts from the relative path **'src'** folder and we are trying to patch outside of it.

By default bitbake patches the files with patch level 'p1' so creating a patch file which matches destination folder structure would solve this issue. Another option is to alter the patch level. E.g.

**SRC_URI += file://docsis_3383.patch;striplevel=0**
**Issue:**

Bitbake complains about md5sum mismatch when a recipe has retained old md5sum value while the source file is updated.

**Example console log:**

ERROR: gst-plugins-playersinkbin-noop: md5 data is not matching for file://gstplayersinkbin.c;md5=0f518921aef846c156f91ce4dd6b7c76
ERROR: gst-plugins-playersinkbin-noop: The new md5 checksum is 958142c8f2783c6c4f357f561585b4da

**Possible solution:**

Update the new md5sum value of the file in recipe. This can be done using following steps:

:…/meta-rdk/recipes-extended/gst-plugins-playersinkbin/files$ md5sum -t gstplayersinkbin.c
958142c8f2783c6c4f357f561585b4da gstplayersinkbin.c

Now change the above value in recipe:

LIC_FILES_CHKSUM = "file://gstplayersinkbin.c;md5=958142c8f2783c6c4f357f561585b4da \"
**Example Console Log :**

repo: warning: Python 2 is no longer supported; Please upgrade to Python 3.6+.
Downloading Repo source from https://gerrit.googlesource.com/git-repo
remote: Finding sources: 100% (32/32)
remote: Total 32 (delta 14), reused 32 (delta 14)
Unpacking objects: 100% (32/32), done.
File "/mnt/home /cmf/.repo/repo/main.py", line 79
file=sys.stderr)
^
SyntaxError: invalid syntax

If you're using an older system without Python 3.6+, try downloading an older version of the Repo Launcher that still supports Python 2.7.

**Possible Solution :**

# create a bin directory

mkdir ~/bin

export PATH=~/bin:$PATH

curl https://storage.googleapis.com/git-repo-downloads/repo-1 > ~/bin/repo

chmod a+x ~/bin/repo

```
ERROR: trower-base64-git+AUTOINC+fbb9440ae2-r0 do_fetch: Fetcher failure: Unable to find revision
fbb9440ae2bc1118866baefcea7ff814f16613dd in branch master even from upstream

ERROR: trower-base64-git+AUTOINC+fbb9440ae2-r0 do_fetch: Fetcher failure for URL: 'git://github.com/Comcast
/trower-base64.git'. Unable to fetch URL from any source.

ERROR: trower-base64-git+AUTOINC+fbb9440ae2-r0 do_fetch: Function failed: base_do_fetch

ERROR: Logfile of failure stored in: /builds/__repo/build-brcm968360GW/tmp/work/cortexa7t2-vfp-rdk-linux-gnueabi
/trower-base64/git+AUTOINC+fbb9440ae2-r0/temp/log.do_fetch.18310

ERROR: Task (/builds/__repo/meta-rdk-ext/recipes-support/trower-base64/trower-base64_1.0.bb:do_fetch) failed with
exit code '1
```

Few points to check here,

- First check if the SRC_URL can be accessed manually in browser
- Check the path of the URL is proper and check the branch detail as well

For example, above error can be fixed by appending **branch=main** in the SRC_URL path