kas Documentation

Release 2.4

Daniel Wagner, Jan Kiszka, Claudius Heine

Feb 25, 2021

Contents

Contents:

CHAPTER 1

Introduction

This tool provides an easy mechanism to setup bitbake based projects.

The OpenEmbedded tooling support starts at step 2 with bitbake. The downloading of sources and then configuration has to be done by hand. Usually, this is explained in a README. Instead kas is using a project configuration file and does the download and configuration phase.

Key features provided by the build tool:

- clone and checkout bitbake layers
- create default bitbake settings (machine, arch, . . .)
- launch minimal build environment, reducing risk of host contamination
- initiate bitbake build process

CHAPTER 2

User Guide

2.1 Dependencies & installation

This project depends on

- Python 3
- distro Python 3 package
- jsonschema Python 3 package
- PyYAML Python 3 package (optional, for yaml file support)

To install kas into your python site-package repository, run:

```
$ sudo pip3 install .
```

2.2 Usage

There are (at least) four options for using kas:

- Install it locally via pip to get the `kas` command.
- Use the container image locally. In this case, download the `kas-container` script from the kas repository and use it in place of the `kas` command. The script version corresponds to the kas tool and the kas image version.
- Use the container image in CI. Specify `ghcr.io/siemens/kas/kas[-isar][:<x.y>]` in your CI script that requests a container image as runtime environment. See https://github.com/orgs/siemens/packages/container/kas%2Fkas/31765 and https://github.com/orgs/siemens/packages/container/kas%2Fkas-isar/31794 for all available images.
- Use the **run-kas** wrapper from this directory. In this case, replace `kas` in the examples below with `path/to/run-kas`.

Start build:

```
$ kas build /path/to/kas-project.yml
```

Alternatively, experienced bitbake users can invoke usual **bitbake** steps manually, e.g.:

```
$ kas shell /path/to/kas-project.yml -c 'bitbake dosfsutils-native'
```

kas will place downloads and build artifacts under the current directory when being invoked. You can specify a different location via the environment variable *KAS_WORK_DIR*.

2.3 Use Cases

1. Initial build/setup:

   ```
   $ mkdir $PROJECT_DIR
   $ cd $PROJECT_DIR
   $ git clone $PROJECT_URL meta-project
   $ kas build meta-project/kas-project.yml
   ```

2. Update/rebuild:

   ```
   $ cd $PROJECT_DIR/meta-project
   $ git pull
   $ kas build kas-project.yml
   ```

2.4 Plugins

kas sub-commands are implemented by a series of plugins. Each plugin typically provides a single command.

2.4.1 `build` plugin

This plugin implements the `kas build` command.

When this command is executed, kas will checkout repositories, setup the build environment and then invoke bitbake to build the targets selected in the chosen config file.

For example, to build the configuration described in the file `kas-project.yml` you could run:

```
kas build kas-project.yml
```

2.4.2 `checkout` plugin

This plugin implements the `kas checkout` command.

When this command is executed, kas will checkout repositories and set up the build directory as specified in the chosen config file. This command is useful if you need to inspect the configuration or modify any of the checked out layers before starting a build.

For example, to setup the configuration described in the file `kas-project.yml` you could run:

```
kas checkout kas-project.yml
```

2.4.3 `for-all-repos` plugin

This plugin implements the `kas for-all-repos` command.

When this command is executed, kas will checkout the repositories listed in the chosen config file and then execute a specified command in each repository. It can be used to query the repository status, automate actions such as archiving the layers used in a build or to execute any other required commands.

For example, to print the commit hashes used by each repository used in the file `kas-project.yml` (assuming they are all git repositories) you could run:

```
kas for-all-repos kas-project.yml 'git rev-parse HEAD'
```

The environment for executing the command in each repository is extended to include the following variables:

- `KAS_REPO_NAME`: The name of the current repository determined by either the name property or by the key used for this repo in the config file.
- `KAS_REPO_PATH`: The path of the local directory where this repository is checked out, relative to the directory where `kas` is executed.
- `KAS_REPO_URL`: The URL from which this repository was cloned, or an empty string if no remote URL was given in the config file.
- `KAS_REPO_REFSPEC`: The refspec which was checked out for this repository, or an empty string if no refspec was given in the config file.

2.4.4 `shell` plugin

This plugin implements the `kas shell` command.

When this command is executed, kas will checkout repositories, setup the build environment and then start a shell in the build environment. This can be used to manually run `bitbake` with custom command line options or to execute other commands such as `runqemu`.

For example, to start a shell in the build environment for the file `kas-project.yml` you could run:

```
kas shell kas-project.yml
```

Or to invoke qemu to test an image which has been built:

```
kas shell kas-project.yml -c 'runqemu'
```

2.5 Project Configuration

Currently, JSON and YAML are supported as the base file formats. Since YAML is arguably easier to read, this documentation focuses on the YAML format.

```
# Every file needs to contain a header, that provides kas with information
# about the context of this file.
header:
  # The `version` entry in the header describes for which configuration
```

(continues on next page)

(continued from previous page)

```
  # format version this file was created for. It is used by kas to figure
  # out if it is compatible with this file. The version is an integer that
  # is increased on every format change.
  version: x
# The machine as it is written into the `local.conf` of bitbake.
machine: qemux86-64
# The distro name as it is written into the `local.conf` of bitbake.
distro: poky
repos:
  # This entry includes the repository where the config file is located
  # to the bblayers.conf:
  meta-custom:
  # Here we include a list of layers from the poky repository to the
  # bblayers.conf:
  poky:
    url: "https://git.yoctoproject.org/git/poky"
    refspec: 89e6c98d92887913cadf06b2adb97f26cde4849b
    layers:
      meta:
      meta-poky:
      meta-yocto-bsp:
```

A minimal input file consists out of the `header`, `machine`, `distro`, and `repos`.

Additionally, you can add `bblayers_conf_header` and `local_conf_header` which are strings that are added to the head of the respective files (`bblayers.conf` or `local.conf`):

```
bblayers_conf_header:
  meta-custom: |
    POKY_BBLAYERS_CONF_VERSION = "2"
    BBPATH = "${TOPDIR}"
    BBFILES ?= ""
local_conf_header:
  meta-custom: |
    PATCHRESOLVE = "noop"
    CONF_VERSION = "1"
    IMAGE_FSTYPES = "tar"
```

`meta-custom` in these examples should be a unique name (in project scope) for this configuration entries. We assume that your configuration file is part of a `meta-custom` repository/layer. This way its possible to overwrite or append entries in files that include this configuration by naming an entry the same (overwriting) or using an unused name (appending).

2.5.1 Including in-tree configuration files

It's currently possible to include kas configuration files from the same repository/layer like this:

```
header:
  version: x
  includes:
    - base.yml
    - bsp.yml
    - product.yml
```

The specified files are addressed relative to your current configuration file.

2.5.2 Including configuration files from other repos

It's also possible to include configuration files from other repos like this:

```
header:
  version: x
  includes:
    - repo: poky
      file: kas-poky.yml
    - repo: meta-bsp-collection
      file: hw1/kas-hw-bsp1.yml
    - repo: meta-custom
      file: products/product.yml
repos:
  meta-custom:
  meta-bsp-collection:
    url: "https://www.example.com/git/meta-bsp-collection"
    refspec: 3f786850e387550fdab836ed7e6dc881de23001b
    layers:
      # Additional to the layers that are added from this repository
      # in the hw1/kas-hw-bsp1.yml, we add here an additional bsp
      # meta layer:
      meta-custom-bsp:
  poky:
    url: "https://git.yoctoproject.org/git/poky"
    refspec: 89e6c98d92887913cadf06b2adb97f26cde4849b
    layers:
      # If `kas-poky.yml` adds the `meta-yocto-bsp` layer and we
      # do not want it in our bblayers for this project, we can
      # overwrite it by setting:
      meta-yocto-bsp: exclude
```

The files are addressed relative to the git repository path.

The include mechanism collects and merges the content from top to bottom and depth first. That means that settings in one include file are overwritten by settings in a latter include file and entries from the last include file can be overwritten by the current file. While merging all the dictionaries are merged recursively while preserving the order in which the entries are added to the dictionary. This means that `local_conf_header` entries are added to the `local.conf` file in the same order in which they are defined in the different include files. Note that the order of the configuration file entries is not preserved within one include file, because the parser creates normal unordered dictionaries.

2.5.3 Including configuration files via the command line

When specifying the kas configuration file on the command line, additional configurations can be included ad-hoc:

```
$ kas build kas-base.yml:debug-image.yml:board.yml
```

This is equivalent to static inclusion from some kas-combined.yml like this:

```
header:
  version: x
  includes:
    - kas-base.yml
    - debug.image.yml
    - board.yml
```

Command line inclusion allows to create configurations on-demand, without the need to write a kas configuration file for each possible combination.

Note that all configuration files combined via the command line either have to come from the same repository or have to live outside of any versioning control. kas will refuse any other combination in order to avoid complications and configuration flaws that can easily emerge from them.

2.5.4 Configuration reference

- **`header`: dict [required]** The header of every kas configuration file. It contains information about the context of the file.
  - **`version`: integer [required]** Lets kas check if it is compatible with this file. See the *configuration format changelog* for the format history and the latest available version.
  - **`includes`: list [optional]** A list of configuration files this current file is based on. They are merged in order they are stated. So a latter one could overwrite settings from previous files. The current file can overwrite settings from every included file. An item in this list can have one of two types:
    - **item: string** The path to a kas configuration file, relative to the current file.
    - **item: dict** If files from other repositories should be included, choose this representation.
      - **`repo`: string [required]** The id of the repository where the file is located. The repo needs to be defined in the `repos` dictionary as `<repo-id>`.
      - **`file`: string [required]** The path to the file relative to the root of the repository.
- **`build_system`: string [optional]** Defines the bitbake-based build system. Known build systems are `openembedded` (or `oe`) and `isar`. If set, this restricts the search of kas for the init script in the configured repositories to `oe-init-build-env` or `isar-init-build-env`, respectively. If `kas-container` finds this property in the top-level kas configuration file (includes are not evaluated), it will automatically select the required container image and invocation mode.
- **`defaults`: dict [optional]** This key can be used to set default values for various properties. This may help you to avoid repeating the same property assignment in multiple places if, for example, you wish to use the same refspec for all repositories.
  - **`repos`: dict [optional]** This key can contain default values for some repository properties. If a default value is set for a repository property it may still be overridden by setting the same property to a different value in a given repository.
    - **`refspec`: string [optional]** Sets the default `refspec` property applied to all repositories that do not override this.
    - **`patches`: dict [optional]** This key can contain default values for some repository patch properties. If a default value is set for a patch property it may still be overridden by setting the same property to a different value in a given patch.
      - **`repo`: string [optional]** Sets the default `repo` property applied to all repository patches that do not override this.
- **`machine`: string [optional]** Contains the value of the `MACHINE` variable that is written into the `local.conf`. Can be overwritten by the `KAS_MACHINE` environment variable and defaults to `qemux86-64`.
- **`distro`: string [optional]** Contains the value of the `DISTRO` variable that is written into the `local.conf`. Can be overwritten by the `KAS_DISTRO` environment variable and defaults to `poky`.
- **`target`: string [optional] or list [optional]** Contains the target or a list of targets to build by bitbake. Can be overwritten by the `KAS_TARGET` environment variable and defaults to `core-image-minimal`. Space is used as a delimiter if multiple targets should be specified via the environment variable.

- **env: dict [optional]** Contains environment variable names with the default values. These variables are made available to bitbake via `BB_ENV_EXTRAWHITE` and can be overwritten by the variables of the environment in which kas is started.
- **task: string [optional]** Contains the task to build by bitbake. Can be overwritten by the `KAS_TASK` environment variable and defaults to `build`.
- **repos: dict [optional]** Contains the definitions of all available repos and layers.
  - **<repo-id>: dict [optional]** Contains the definition of a repository and the layers, that should be part of the build. If the value is `None`, the repository, where the current configuration file is located is defined as `<repo-id>` and added as a layer to the build.
    - **name: string [optional]** Defines under which name the repository is stored. If its missing the `<repo-id>` will be used.
    - **url: string [optional]** The url of the repository. If this is missing, no version control operations are performed.
    - **type: string [optional]** The type of version control repository. The default value is `git` and `hg` is also supported.
    - **refspec: string [optional]** The refspec that should be used. If `url` was specified but no `refspec` the revision you get depends on the defaults of the version control system used.
    - **path: string [optional]** The path where the repository is stored. If the `url` and `path` is missing, the repository where the current configuration file is located is defined. If the `url` is missing and the path defined, this entry references the directory the path points to. If the `url` as well as the `path` is defined, the path is used to overwrite the checkout directory, that defaults to `kas_work_dir` + `repo.name`. In case of a relative path name `kas_work_dir` is prepended.
    - **layers: dict [optional]** Contains the layers from this repository that should be added to the `bblayers.conf`. If this is missing or `None` or and empty dictionary, the path to the repo itself is added as a layer.
      - **<layer-path>: enum [optional]** Adds the layer with `<layer-path>` that is relative to the repository root directory, to the `bblayers.conf` if the value of this entry is not in this list: `['disabled', 'excluded', 'n', 'no', '0', 'false']`. This way it is possible to overwrite the inclusion of a layer in latter loaded configuration files.
    - **patches: dict [optional]** Contains the patches that should be applied to this repo before it is used.
      - **<patches-id>: dict [optional]** One entry in patches with its specific and unique id. All available patch entries are applied in the order of their sorted `<patches-id>`.
      - **repo: string [required]** The identifier of the repo where the path of this entry is relative to.
      - **path: string [required]** The path to one patch file or a quilt formatted patchset directory.
- **bblayers_conf_header: dict [optional]** This contains strings that should be added to the `bblayers.conf` before any layers are included.
  - **<bblayers-conf-id>: string [optional]** A string that is added to the `bblayers.conf`. The entry id (`<bblayers-conf-id>`) should be unique if lines should be added and can be the same from another included file, if this entry should be overwritten. The lines are added to `bblayers.conf` in the same order as they are included from the different configuration files.
- **local_conf_header: dict [optional]** This contains strings that should be added to the `local.conf`.
  - **<local-conf-id>: string [optional]** A string that is added to the `local.conf`. It operates in the same way as the `bblayers_conf_header` entry.

- **`proxy_config`: dict [optional]** Defines the proxy configuration bitbake should use. Every entry can be overwritten by the respective environment variables.
  - `http_proxy`: string [optional]
  - `https_proxy`: string [optional]
  - `no_proxy`: string [optional]

CHAPTER 3

Command line usage

kas - setup tool for bitbake based project

```
usage: kas [-h] [--version] [-d] {build,checkout,for-all-repos,shell} ...
```

3.1 Positional Arguments

cmd Possible choices: build, checkout, for-all-repos, shell

sub command help

3.2 Named Arguments

--version show program's version number and exit

-d, --debug Enable debug logging

Default: False

3.3 Sub-commands:

3.3.1 build

Checks out all necessary repositories and builds using bitbake as specified in the configuration file.

```
kas build [-h] [--skip SKIP] [--force-checkout] [--update] [--target TARGET]
          [-c TASK]
          config [extra_bitbake_args [extra_bitbake_args ...]]
```

Positional Arguments

config Config file

extra_bitbake_args Extra arguments to pass to bitbake

Named Arguments

--skip Skip build steps

Default: []

--force-checkout Always checkout the desired refspec of each repository, discarding any local changes

Default: False

--update Pull new upstream changes to the desired refspec even if it is already checked out locally

Default: False

--target Select target to build

-c, --cmd, --task Select which task should be executed

3.3.2 checkout

Checks out all necessary repositories and sets up the build directory as specified in the configuration file.

```
kas checkout [-h] [--skip SKIP] [--force-checkout] [--update] config
```

Positional Arguments

config Config file

Named Arguments

--skip Skip build steps

Default: []

--force-checkout Always checkout the desired refspec of each repository, discarding any local changes

Default: False

--update Pull new upstream changes to the desired refspec even if it is already checked out locally

Default: False

3.3.3 for-all-repos

Runs a specified command in all checked out repositories.

```
kas for-all-repos [-h] [--skip SKIP] [--force-checkout] [--update]
                  config command
```

Positional Arguments

config Config file

command Command to be executed as a string.

Named Arguments

--skip Skip build steps

Default: []

--force-checkout Always checkout the desired refspec of each repository, discarding any local changes

Default: False

--update Pull new upstream changes to the desired refspec even if it is already checked out locally

Default: False

3.3.4 shell

Run a shell in the build environment.

```
kas shell [-h] [--skip SKIP] [--force-checkout] [--update] [-k] [-c COMMAND]
          config
```

Positional Arguments

config Config file

Named Arguments

--skip Skip build steps

Default: []

--force-checkout Always checkout the desired refspec of each repository, discarding any local changes

Default: False

--update Pull new upstream changes to the desired refspec even if it is already checked out locally

Default: False

-k, --keep-config-unchanged Skip steps that change the configuration

Default: False

-c, --command Run command

Default: “”

3.4 Environment variables

Environment variables	Description
KAS_WORK_DIR	The path of the kas work directory, current work directory is the default.
KAS_REPO_REF_DIR	The path to the repository reference directory. Repositories in this directory are used as references when cloning. In order for kas to find those repositories, they have to be named in a specific way. The repo URLs are translated like this: “https://github.com/siemens/meta-iot2000.git” resolves to the name “github.com.siemens.meta-iot2000.git”.
KAS_DISTRO KAS_MACHINE KAS_TARGET KAS_TASK	This overwrites the respective setting in the configuration file.
KAS_PREMIRRORS	Specifies alternatives for repo URLs. Just like bitbake PREMIRRORS, this variable consists of new-line separated entries. Each entry defines a regular expression to match a URL and, space- separated, its replacement. E.g.: “https://.*.somehost.io/ https://localmirror.net/”
SSH_PRIVATE_KEY	Path to the private key file that should be added to an internal ssh-agent. This key cannot be password protected. This setting is useful for CI build servers. On desktop machines, an ssh-agent running outside the kas environment is more useful.
SSH_AUTH_SOCK	SSH authentication socket. Used for cloning over SSH (alternative to SSH_PRIVATE_KEY).
DL_DIR SSTATE_DIR TMPDIR	Environment variables that are transferred to the bitbake environment.
http_proxy https_proxy ftp_proxy no_proxy	This overwrites the proxy configuration in the configuration file.
GIT_PROXY_COMMAND NO_PROXY	Set proxy for native git fetches. NO_PROXY is evaluated by OpenEmbedded’s oe-git-proxy script.
SHELL	The shell to start when using the *shell* plugin.
TERM	The terminal options used in the *shell* plugin.
AWS_CONFIG_FILE AWS_SHARED_CREDENTIALS_FILE	Path to the awscli configuration and credentials file that are copied to the kas home dir.

CHAPTER 4

Developer Guide

4.1 Deploy for development

This project uses pip to manage the package. If you want to work on the project yourself you can create the necessary links via:

```
$ pip3 install --user -e .
```

That will install a backlink ~/.local/bin/kas to this project. Now you are able to call it from anywhere.

4.2 Docker image build

Just run:

```
$ docker build -t <image_name> .
```

When you need a proxy to access the internet, add:

```
--build-arg http_proxy=<http_proxy> --build-arg https_proxy=<https_proxy> --build-arg␣
↪ftp_proxy=<ftp_proxy> --build-arg no_proxy=<no_proxy>
```

to the call.

4.3 Community Resources

Project home:

- https://github.com/siemens/kas

Source code:

- https://github.com/siemens/kas.git
- git@github.com:siemens/kas.git

Documentation:

- https://kas.readthedocs.org

Mailing list:

- kas-devel@googlegroups.com
- Subscription:
  - kas-devel+subscribe@googlegroups.com
  - https://groups.google.com/forum/#!forum/kas-devel/join
- Archives
  - https://groups.google.com/forum/#!forum/kas-devel
  - https://www.mail-archive.com/kas-devel@googlegroups.com/

4.4 Class reference documentation

4.4.1 `kas.kas` Module

This module is the main entry point for kas, setup tool for bitbake based projects

`kas.kas.`**`create_logger`**`()`
Setup the logging environment

`kas.kas.`**`interruption`**`()`
Ignore SIGINT/SIGTERM in kas, let them be handled by our sub-processes

`kas.kas.`**`kas`**`(`*argv*`)`
The actual main entry point of kas.

`kas.kas.`**`kas_get_argparser`**`()`
Creates an argparser for kas with all plugins.

`kas.kas.`**`main`**`()`
The main function that operates as a wrapper around kas.

4.4.2 `kas.libkas` Module

This module contains the core implementation of kas.

`class` `kas.libkas.`**`LogOutput`**`(`*live*`)`
Handles the log output of executed applications

`log_stderr``(`*line*`)`
This method is called when a line is received over stderr.

`log_stdout``(`*line*`)`
This method is called when a line is received over stdout.

`kas.libkas.`**`find_program`**`(`*paths*, *name*`)`
Find a file within the paths array and returns its path.

kas.libkas.**get_build_environ**(*build_system*)
Creates the build environment variables.

kas.libkas.**repos_apply_patches**(*repos*)
Applies the patches to the repositories.

kas.libkas.**repos_fetch**(*repos*)
Fetches the list of repositories to the kas_work_dir.

kas.libkas.**run_cmd**(*cmd*, *cwd*, *env=None*, *fail=True*, *liveupdate=True*)
Runs a command synchronously.

kas.libkas.**run_cmd_async**(*cmd*, *cwd*, *env=None*, *fail=True*, *liveupdate=True*)
Run a command asynchronously.

kas.libkas.**ssh_add_key**(*env*, *key*)
Adds an ssh key to the ssh-agent

kas.libkas.**ssh_cleanup_agent**()
Removes the identities and stops the ssh-agent instance

kas.libkas.**ssh_no_host_key_check**()
Disables ssh host key check

kas.libkas.**ssh_setup_agent**(*envkeys=None*)
Starts the ssh-agent

4.4.3 kas.libcmds Module

This module contains common commands used by kas plugins.

class kas.libcmds.**CleanupSSHAgent**
Removes all the identities and stops the ssh-agent instance.

execute(*ctx*)
This method executes the command.

class kas.libcmds.**Command**
An abstract class that defines the interface of a command.

execute(*ctx*)
This method executes the command.

class kas.libcmds.**FinishSetupRepos**
Finalizes the repo setup loop

execute(*ctx*)
TODO refactor protected-access

class kas.libcmds.**InitSetupRepos**
Prepares setting up repos including the include logic

execute(*ctx*)
This method executes the command.

class kas.libcmds.**Loop**(*name*)
A class that defines a set of commands as a loop.

add(*command*)
Appends a command to the loop.

execute(*ctx*)
Executes the loop.

class kas.libcmds.**Macro**(*use_common_setup=True*, *use_common_cleanup=True*)
Contains commands and provides method to run them.

add(*command*)
Appends commands to the command list.

run(*ctx*, *skip=None*)
Runs a command from the command list with respect to the configuration.

class kas.libcmds.**ReposApplyPatches**
Applies the patches defined in the configuration to the repositories.

execute(*ctx*)
This method executes the command.

class kas.libcmds.**ReposCheckout**
Ensures that the right revision of each repo is checked out.

execute(*ctx*)
This method executes the command.

class kas.libcmds.**ReposFetch**
Fetches repositories defined in the configuration

execute(*ctx*)
This method executes the command.

class kas.libcmds.**SetupDir**
Creates the build directory.

execute(*ctx*)
This method executes the command.

class kas.libcmds.**SetupEnviron**
Sets up the kas environment.

execute(*ctx*)
This method executes the command.

class kas.libcmds.**SetupHome**
Sets up the home directory of kas.

execute(*ctx*)
This method executes the command.

class kas.libcmds.**SetupReposStep**
Single step of the checkout repos loop

execute(*ctx*)
TODO refactor protected-access

class kas.libcmds.**SetupSSHAgent**
Sets up the ssh agent configuration.

execute(*ctx*)
This method executes the command.

class kas.libcmds.**WriteBBConfig**
Writes bitbake configuration files into the build directory.

execute(*ctx*)
This method executes the command.

4.4.4 kas.config Module

This module contains the implementation of the kas configuration.

class kas.config.**Config**(*filename*, *target=None*, *task=None*)
Implements the kas configuration based on config files.

find_missing_repos()
Returns repos that are in config but not on disk

get_bblayers_conf_header()
Returns the bblayers.conf header

get_bitbake_targets()
Returns a list of bitbake targets

get_bitbake_task()
Returns the bitbake task

get_build_system()
Returns the pre-selected build system

get_distro()
Returns the distro

get_environment()
Returns the configured environment variables from the configuration file with possible overwritten values from the environment.

get_local_conf_header()
Returns the local.conf header

get_machine()
Returns the machine

get_multiconfig()
Returns the multiconfig array as bitbake string

get_repos()
Returns the list of repos.

4.4.5 kas.repos Module

This module contains the Repo class.

class kas.repos.**GitRepo**(*name*, *url*, *path*, *refspec*, *layers*, *patches*, *disable_operations*)
Provides the git functionality for a Repo.

class kas.repos.**MercurialRepo**(*name*, *url*, *path*, *refspec*, *layers*, *patches*, *disable_operations*)
Provides the hg functionality for a Repo.

class kas.repos.**Repo**(*name*, *url*, *path*, *refspec*, *layers*, *patches*, *disable_operations*)
Represents a repository in the kas configuration.

static factory(*name*, *repo_config*, *repo_defaults*, *repo_fallback_path*)
Returns a Repo instance depending on params.

static get_root_path(*path*, *fallback=True*)
Checks if path is under version control and returns its root path.

class kas.repos.**RepoImpl**(*name*, *url*, *path*, *refspec*, *layers*, *patches*, *disable_operations*)
Provides a generic implementation for a Repo.

apply_patches_async()
Applies patches to a repository asynchronously.

checkout()
Checks out the correct revision of the repo.

fetch_async()
Starts asynchronous repository fetch.

4.4.6 kas.includehandler Module

This module implements how includes of configuration files are handled in kas.

exception kas.includehandler.**IncludeException**
Class for exceptions that appear in the include mechanism.

class kas.includehandler.**IncludeHandler**(*top_files*)
Implements a handler where every configuration file should contain a dictionary as the base type with and 'includes' key containing a list of includes.

The includes can be specified in two ways: as a string containing the relative path from the current file or as a dictionary. The dictionary should have a 'file' key containing the relative path to the include file and optionally a 'repo' key containing the key of the repository. If the 'repo' key is missing the value of the 'file' key, it is treated the same as if just a string was defined, meaning the path is relative to the current config file. Otherwise it is interpreted relative to the repository path.

The includes are read and merged from the deepest level upwards.

get_config(*repos=None*)

Parameters: repos – A dictionary that maps repo names to directory paths

Returns:

(config, repos) config – A dictionary containing the configuration repos – A list of missing repo names that are needed to create a complete configuration

exception kas.includehandler.**LoadConfigException**(*message*, *filename*)
Class for exceptions that appear while loading a configuration file.

kas.includehandler.**load_config**(*filename*)
Load the configuration file and test if version is supported.

4.4.7 kas.plugins Module

This module contains and manages kas plugins

kas.plugins.**all**()
Get a list of all loaded kas plugin classes

kas.plugins.**get**(*name*)
Lookup a kas plugin class by name

kas.plugins.**load**()
Import all kas plugins

kas.plugins.**register_plugins**(*mod*)
Register all kas plugins found in a module

CHAPTER 5

Configuration Format Changes

5.1 Version 1 (Alias '0.10')

5.1.1 Added

- Include mechanism
- Version check

5.2 Version 2

5.2.1 Changed

- Configuration file versions are now integers

5.2.2 Fixed

- Including files from repos that are not defined in the current file

5.3 Version 3

5.3.1 Added

- `Task` key that allows to specify which task to run (`bitbake -c`)

5.4 Version 4

5.4.1 Added

- `Target` key now allows to be a list of target names

5.5 Version 5

5.5.1 Changed behavior

- Using `multiconfig:*` targets adds appropriate `BBMULTICONFIG` entries to the `local.conf` automatically.

5.6 Version 6

5.6.1 Added

- `env` key now allows to pass custom environment variables to the bitbake build process.

5.7 Version 7

5.7.1 Added

- `type` property to `repos` to be able to express which version control system to use.

5.8 Version 8

5.8.1 Added

- `patches` property to `repos` to be able to apply additional patches to the repo.

5.9 Version 9

5.9.1 Added

- `defaults` key can now be used to set a default value for the repository property `refspec` and the repository patch property `repo`. These default values will be used if the appropriate properties are not defined for a given repository or patch.

5.10 Version 10

5.10.1 Added

- `build_system` property to pre-select OE or Isar.

CHAPTER 6

Indices and tables

- genindex
- modindex
- search

Python Module Index

k

Index

L

M

R

S

W