Table of Contents

The AtlasSetup environment setup package....1

1. Introduction....2
1.1. Location of releases....2

2. Customizing your login shell....3
2.1 Configuration using AtlasSetup environment variable....3
2.1.1 Using asetup on CVMFS....3
2.1.2 Using asetup on CERN AFS....3
2.2 Quick start with some asetup examples....3
2.2.1 Example of release setup....3
2.2.1 Example of non-release env setup....4

3. Executing the asetup command....5
3.1 asetup command line help....5

4. AtlasSetup Configuration Files and override priorities....8
4.1. Short example of an asetup configuration file....9

5. Test release directory names....11

6. Testing of new AtlasSetup on the Grid....12

7. Platforms and Compilers....13

8. Environment variables setup by AtlasSetup....14
8.1 Runtime and CMT-related environment variables....14
8.2 User and release related environment variables....14

9. Pitfalls and differences relative to the use of AtlasLogin....15

The AtlasSetup environment setup package

This is a brief users guide. For full details, refer to the AtlasSetupReference page.

1. Introduction

The `AtlasSetup` package is designed to allow both users and developers to setup an environment for working with ATLAS releases. This is a brief users guide. For full details, refer to the AtlasSetupReference page.

This documentation corresponds to `AtlasSetup-00-08-02`.

Please report any suspected problems to Shuwei Ye.

1.1. Location of releases

At CERN, releases are located in several different areas:

In the default configuration, AtlasSetup will locate numbered releases (e.g. `20.20.3`) in the `releases` area and nightly builds (e.g. `21.0.X/2017-02-20T2119`) in the `nightlies` area. This is appropriate for the CERN configuration. Several such areas are supported by AtlasSetup:

- The `releases` area (on cvmfs by default), containing numbered releases (e.g. `20.20.3`) that are installed from distribution kits. This area is designed to act as a remote site (albeit located at CERN), similarly to the many other remote sites in the ATLAS computing grid. It is expected that users of the ATLAS software will use these releases. The actual location is determined by the --releasesarea <dirs> option.
- The `builds` area, containing the directly built numbered releases that are used as a stepping stone towards the final releases in the `releases` area that are installed from the distribution kits. These releases are expected to be only of use to software developers, and not for normal users. Their actual location is determined by the --buildsarea <dirs> option.
- The `nightlies` area (on cvmfs by default), containing the nightly builds. These releases are expected to be only of use to software developers, and not for normal users. Their actual location is determined by the --nightliesarea <dirs> option.
- The `tdaq` area, containing TDAQ project releases. Their actual location is determined by the --tdaqarea <dirs> option.
- The `externals` area, containing LCGCMT and some of the older GAUDI releases. Their actual location is determined by the --externalsarea <dirs> option.

In order to provide transparent access to releases in these locations at CERN, but also to provide easy access to releases installed at remote sites, AtlasSetup uses the concept of a *search path* in order to search for releases in possibly multiple locations. Thus each of the above areas may be specified as a list of directories (separated by comma or colon characters). By default it will look in the areas in the order `releases`, `builds`, `nightlies` etc., but this default may be restricted to only one area.

In the default configuration, AtlasSetup will locate numbered releases (e.g. `20.20.3`) in the `releases` area and nightly builds (e.g. `21.0.X/2017-02-20T2119`) in the `nightlies` area.

2. Customizing your login shell

In order to access the python-based AtlasSetup scripts (which will be referred to as `asetup` for the remainder of this document), it's necessary to configure your login environment based on a knowledge of the location of one release at your site (or at CERN AFS if you have access to that). Apart from that, it should not be necessary for you to perform any site-specific configuration, but several detailed configuration overrides and customisations can be specified either on the command line, or in configuration files that you can create and manage.

The following assumes that you know the location of at least one release at your site, which is referred to as `<path-to-release>` in the following description. There are two variants of the procedure, one of which reproduces the actions of the ATLAS production system and Ganga and pAthena. The other is more suited for interactive use since it involves a one-time modification to your login shell (e.g. `${HOME}/.bashrc` or `${HOME}/.tcshrc`) , although either can be used interchangeably.

2.1 Configuration using `AtlasSetup` environment variable

2.1.1 Using asetup on CVMFS

The alias/command **asetup** should be available after having run **setupATLAS** if you have CVMFS. If you do not have the alias **setupATLAS** defined, you can set up ALRB env (replace `.sh` with `.csh` for csh):

```
export ATLAS_LOCAL_ROOT_BASE=/cvmfs/atlas.cern.ch/repo/ATLASLocalRootBase
source $ATLAS_LOCAL_ROOT_BASE/user/atlasLocalSetup.sh
```

In case of using the beta version of asetup on cvmfs, you need define `export ALRB_asetupVersion=testing` (`setenv ALRB_asetupVersion=testing` for csh) prior to running **setupATLAS** or sourcing `atlasLocalSetup.sh`.

2.1.2 Using asetup on CERN AFS

Users are recommended to use asetup on cvmfs, but if you like to asetup on CERN AFS for any reason, you can set up:

```
export AtlasSetup="/afs/cern.ch/atlas/software/dist/AtlasSetup"
alias asetup='source $AtlasSetup/scripts/asetup.sh'
```

For the beta version, just replace the **AtlasSetup** with `/afs/cern.ch/atlas/software/dist/beta/AtlasSetup`.

2.2 Quick start with some asetup examples

Now you have defined command **asetup** in the previous section. You can find some examples in this section.

You can get the asetup version by running `asetup --version`, which will print `AtlasSetup-00-08-02` for example.

2.2.1 Example of release setup

If you want to setup the runtime environment for the `AtlasProduction 20.20.3.2` release just type:

```
asetup 20.20.3.2
```

This will automatically setup the runtime for `AtlasProduction 20.20.3.2` from the `releases` area at CERN.

```
asetup AnalysisBase,21.2,latest
```

This will set up the **latest** nightlies release of 21.2 for `AnalysisBase`.

If you are working at a single site, it's not normally necessary to override any site-specific defaults when configuring AtlasSetup since those should already be setup by the kit installation procedure (with the possibility of a site administrator being able to add further customisation).

An example of a user configuration file is given in Section 4.1.

2.2.1 Example of non-release env setup

If you would like to set up an env to build a release, you can:

```
asetup cmakesetup,none,gcc62
```

In the above example, `asetup` would use the tag **gcc62** to extract gcc version. You can override the gcc version via option `--gccversion=6.3.0`. You can also specify the cmake version using `--cmakeversion=3.9.6`.

3. Executing the asetup command

The environment setup is performed by executing the asetup command with a list of options and arguments, thus:

```
asetup <arg1> <arg2>,<arg3> <arg4>=<value4> --<option1> --<option2> <value2> --<option3>=<value3>
```

where:

- `<arg1>`, `<arg2>` and `<arg3>` are arguments, which may be separated by spaces or commas (",").
- `<arg4>` is an argument with value `<value4>` which is equivalent to `--<arg4>=<value4>`. Note that if an option is both specified in this manner, and as a command line option with value (`--<arg4>=<value4>`), then the value specified by the command line option syntax takes precedence.
- `<option1>` is an option without value (corresponding to a particular value of a boolean variable).
- `<option2>` and `<option3>` are options with value `<value2>` and `<value3>` respectively, which cannot be defaulted (i.e. a value must be supplied). The option name and its value can be separated by either a space or equals character.
- `--tags` (which has the aliases `-t` or `--tag`) can be used to specify a comma-separated list of tags (corresponding to command line arguments). This syntax is equivalent to specifying the tags as arguments, but is retained for better backwards compatibility with AtlasLogin tags. However, it is recommended that arguments are used instead of this syntax.

Some commonly used options have single character aliases, in which case they are denoted by a single rather than a double dash (e.g. `-t` instead of `-tags`).

Most configuration variables can be specified as arguments, options or tags, although some that have associated values can only be specified as option/value pairs. The command line options, arguments and tags and the corresponding variables are described in the AtlasSetup Reference Manual Section 8.

3.1 asetup command line help

There are about 200 options, you can get the brief help by specifying either of:

```
asetup -h
asetup --help
```

which will print out the following:

```
Usage: asetup.py [options]

Options:
  -h, --help            show this help message and exit
  --cmtconfig=<CONFIG>, --CMTCONFIG=<CONFIG>, --platform=<CONFIG>
                        Force the CONFIG setting
  --nightliesarea=<dir>:<dir2>
                        The nightlies area (which can be a search path)
  --releasesarea=<dir>:<dir2>
                        Override the default location of the releases area
  --siteconfigfile=<file>
                        Specify location of site-specific configuration file
  --testarea=<dir>, --workarea=<dir>
                        Set location of the test/work area, or you can use tag
                        'here' to use PWD as test area
  --dbarea=<dir>        Override the default location of the ATLAS_DB_AREA area
  --dbrelease=<release>
                        The DB release name (e.g. 11.1.2)
  --gcclocation=<dir>, --gcc43location=<dir>
```

```
                        The gcc compiler location (default contrib/gcc)
  --helpMoreOn=<group>  The group name for help printout, valid groups: [runti
                        me,cmake,relLoc,python,asetup,area,db,project,platform
                        ,relName,envVar,cmt,ROOT,other,compiler,All], where
                        'All' will print all groups options

For further details, look at
https://twiki.cern.ch/twiki/bin/view/AtlasComputing/AtlasSetupReference
```

You can make use of the special option **--HelpMoreOn** to get more details on specific group, says, *relName*, by command **asetup --helpMoreOn=relName**:

```
% asetup --helpMoreOn=relName
relName Options:
  Release type/number, branch (nightly) name

  -b <branch>, --branch=<branch>
                        The branch name (e.g. 15.X.0, 15.X.0-MIG5)
  --latest, --latestrelease
                        Use latest nightlies release (default False)
  --lcgnightly=<value>  The LCG nightly to setup (default is <automatic>)
  -r <release>, --release=<release>
                        The release name (e.g. 15.6.5, rel_1, or timestamp
                        nightly like rMM-DD, rDD)
```

Another example for **--helpMoreOn=asetup**,

```
asetup --helpMoreOn=asetup
asetup Options:
  AtlasSetup location and behavior

  --version             Report the AtlasSetup version number
  --setuprelease, --nosetuprelease, --none
                        Perform release setup (default True)
  --simulate            Show me what you would do, but do not actually do it

  --silent              Perform silently, suppressing any error or warning messages
  --warnunassigned, --errorunassigned
                        Unassigned arguments cause warnings or errors (default)
  --asetupafs=<dir>     Location of AtlasSetup on AFS
  --asetupcvmfs=<dir>   Location of AtlasSetup on CVMFS
  --autorestore         Autorestore last configuration if command line is empty
  --buildcache, --cache, --nobuildcache, --nocache
                        Build cached shellfile (default is False)
  --bypass, --bypasssite, --bypasssitesetup
                        Bypass the site configuration setup file
  --debugfunctions=<list>
                        List of functions to generate debug information (or
                        <all>) if --debugprint not enabled
  --debugprint          Debug print mode (default is not debug)
  --dumpconfig, --nodumpconfig
                        Dump the configuration (default False)
  --expandsymlinks, --noexpandsymlinks
                        Expand symlinks when setting up environment (default False)
  -i <inputFile>, --input=<inputFile>, --inputfile=<inputFile>
                        Read configuration from file <inputFile>
  --ignorewarning       Ignore errors of Compiler/Python setup in setuprelease
  -o <outputFile>, --output=<outputFile>, --outputfile=<outputFile>
                        Write configuration to file <ouputFile> (default is
                        standard out)
  --oldwriteshell, --nooldwriteshell, --newwriteshell
                        Use old shell (default) writing method (for debugging)
  --platforms           List available platforms for setup release
  --printlevel=<level>  Specify verbosity print level (default is 0)
  --projtest, --projTest, --notsimple, --notsimpletest
```

```
                      Setup a test area including the project name
--restore             Restore the last saved session in this directory
--save, --nosave      Save the session in this directory (default False)
--shellprolog=<shell-prolog>
                      Name of shell prolog file (default shell_prolog)
--siteconfigfile=<file>
                      Specify location of site-specific configuration file
--slim, --slimpaths, --noslim, --noslimpaths
                      Slim paths by removing nonexistent and empty entries
                      (default False)
-t <tag-list>, --tag=<tag-list>, --tags=<tag-list>
                      The list of tags to setup
--testmode, --notestmode
                      Activate special test mode code (for testing purposes
                      only) (default False)
--unsetprevious, --unset, --nounsetprevious, --nounset
                      Unset environment from previous release setup (default
                      is disabled)
--useenvironment, --environment, --nouseenvironment, --noenvironment
                      Setup release from existing environment (default is disabled)
--user, --userpriority
                      Give the user environment variables priority over
                      AtlasSetup ones
-v <value>, --verbose=<value>, --VERBOSE=<value>
                      Verbose compiler mode True/False/None (default is False)
```

4. AtlasSetup Configuration Files and override priorities

The default configuration can be overridden on a per site, per user, per directory and per usage basis. A hierarchy of possible override files is searched for, where the presence of overrides in each has priority over those established by previous ones, or the overall defaults. The setting of arguments, options or tags using the command line overrides the values that are established by default or via such configuration files.

The following files are searched for:

```
--siteconfigfile <files>  [see notes]
${AtlasSetupSite}
${HOME}/.asetup
${PWD}/.asetup
--input <input-file>
--restore
```

The `siteconfigfile` configuration variable specifies a search path of site configuration files that will be read to override or append to the AtlasSetup intrinsic defaults. The default value is `../cmtsite/asetup_defaults:../.asetup.site`, where relative paths are taken to be relative to the `${AtlasSetup}` environment variable location, and the contents of these files will override or append to the AtlasSetup defaults. The value of this variable may be overridden on the command line using the --siteconfigfile <files> command line option/value pair. If multiple files are specified (as is the case for the default), then subsequent ones override or append to the configuration specified by earlier ones. The action of these files can be disabled for diagnostics purposes by use of the --bypasssite command line option, or by specifying the `<files>` value to the --siteconfigfile <files> command line option/value pair as `""` or `"<none>"`.

If the `$AtlasSetupSite` environment variable exists and is set to the location of a valid configuration file, the contents of the file override or are appended to the AtlasSetup defaults (including any site-specific ones due to the `${AtlasSetup}/../cmtsite/asetup_defaults` file). This feature is designed to allow sites to provide a site-wide configuration file that will be applied by default for all users of that site. The only way to disable the application of such a site-wide configuration is to delete the environment variable, set it to being empty or referencing an empty configuration file. Please note that this environment variable should be set to reference a file with a name of your choice *including the filename itself*.

If the `${HOME}/.asetup` file exists (i.e. in the user's home directory) then it's contents override or are appended to the defaults (including any site-specific ones described above) **unless** the --input <input-file> command line option is specified, in which case this file is ignored.

If the `${PWD}/.asetup` file exists (i.e. in the current directory) then it's contents override or are appended to the defaults **including** those that were established by the `${HOME}/.asetup` file and possible site-specific ones **unless** the --input <input-file> command line option is specified, in which case this file is ignored.

If the --input <input-file> command line option is specified, then the contents of `<input-file>` override or are appended to the defaults, and the `${HOME}/.asetup` and `${PWD}/.asetup` files are ignored. If `--input None` or `--input <none>` are specified, all input file processing will be disabled apart from any site-wide configuration applied as a result of the `$AtlasSetupSite` environment variable or `${AtlasSetup}/../cmtsite/asetup_defaults` file.

The --restore command line option operates in conjunction with the --save command line option, and is typically activated by enabling save mode by default using a configuration file (typically `${HOME}/.asetup`). Once this mode has been setup, AtlasSetup will write a save file (`.asetup.save`) into each directory that it has been executed from, and the user can repeat the last activation from that directory using the --restore

command line option (in which case all other configuration files are ignored apart from a site-wide file based on the `$AtlasSetupSite` environment variable). This mode is automatically deactivated if the current working directory is not writable, in which case a warning message will be printed unless --silent mode is enabled, or the --nosave option is used.

The format of asetup configuration files is described in the Atlas Setup Reference Manual Section 10.2.

4.1. Short example of an asetup configuration file

Here is a short example:

```
[defaults]         # Note 1
save = True    # Note 2
runtime = True    # Note 3
verbose = True    # Note 4
multitest = True    # Note 5
testarea: /afs/cern.ch/user/d/dquarrie/w0/Atlas    # Note 6
briefprint = True   # Note 7

[aliases]    # Note 8
bug    = 15.6.X
bugfix = 15.6.X
bugval = 15.6.X-VAL
repro-dec09 = AtlasProduction,15.5.5.8    # Note 9
devel = testarea=${HOME}/tests/devel:multi
data = testarea=${HOME}/tests/data:single

[environment]        # Note 10
AtlasDist = /afs/cern.ch/atlas/software/dist
AtlasTrials = /afs/cern.ch/atlas/software/dist/trials
```

Notes:

1. The `[defaults]` line can be omitted, in which case subsequent lines will be treated as being the settings of configuration variables until another section header is encountered.
2. Enabling the save option will allow you to later restore the last configuration that was setup for the current working directory.
3. Enabling the runtime variable will cause the full release runtime to be setup by default. In fact this is the overall AtlasSetup default, so this line is not really necessary, but is used in this example for clarity.
4. This line causes the `VERBOSE` environment variable to be set, which will result in more information being reported by make. This is described in the verbose command line argument section.
5. Setting the multi or multitest variable overrides the default of a single test directory being setup, independent of the release.
6. Setting the testarea variable to a fixed directory location overrides the default of no test directory being setup.
7. Setting the brief or briefprint variable causes a brief summary of the release to be reported once it's been setup.
8. The [aliases] section header allows you to define aliases for other variables or groups of variables. In this example, the `bug`, `bugfix`, and `bugval` aliases are aliases for branches, which could have been declared in the `[branch]` section.
9. Aliases defined within the `[aliases]` section can also reference lists of other variables. In this case, the `repro-dec09` alias will automatically setup both the project and release number (Need to get the actual ones!!!). Furthermore, the other variables may have values specified (`<variable>=<value>`). Elements of the list should be separated by comma (",") or semicolon (";") characters (where either can be used but each line should use only one of them), and boolean variables may be specified without a value, corresponding to `True`.

10. The `[environment]` section allows you to define environment variables.

5. Test release directory names

A test release directory or directories containing locally checked out packages may be specified by the user. The default is for no such directory to be specified, but the --testarea <dir> command line option, or the corresponding testarea = <dir> configuration variable may be used to setup a test release. Several other configuration variables and options control whether a single such test release area is created, or whether several, perhaps corresponding to different release numbers, are created. In general it is the users responsibility to create and maintain the contents of such test release areas, but one option will automatically create such an area if it doesn't already exist, and to change the current directory location to it., The following configurations are supported:

The --testarea <dir> command line option and associated configuration file entry specifes that the test directory should be setup in `<dir>`, being a directory location. However, a value of `None` or `<none>` inhibits setting up a test directory altogether (as does the notest command line argument) and a value of `<pwd>` or `$PWD` or ${PWD}= sets the current directory location as being the test release area.

The multi or multitest or single, singletest or onetest command line arguments control whether a subdirectory is setup below the overall test release directory. The default is for none to be setup.

Finally, the projtest or simpletest variable and the testtype <type> variable may be used to specify the format of the subdirectory name when one is setup.

The logic for this is:

- If projtest is set to `True` (or simpletest is set to `False`), the AtlasLogin default test release directory name of `<project>-<release>` (e.g. `AtlasOffline-15.6.8`) is setup.
- If projtest is set to `False` (or simpletest is set to `True`), the value of the testtype <type> option determines the format of the subdirectory name. This defaults to `<release>` and corresponds to the release name (e.g. `15.6.9`, `rel_0`), but can take the form of a string where the following substitutions will be made:
    - `<release>` is replaced by the actual release name (e.g. `15.6.9`, `rel_0`).
    - `<base>` is replaced by the 3-digit release name (e.g. `15.6.9` if the release is `15.6.9.4`) or the release name itself if there is no 3-digit equivalent (e.g. `rel_0` for release `rel_0`).
    - `<project>` is replaced by the project name ((e.g. `AtlasOffline`, `AtlasProduction`).
    - `None` or `none` is equivalent to setting multi or multitest or single, singletest or onetest to `True`.

When a test release area is specified, the autocdtest variable may be specified to `True`, in which case the test release directory will be created it if doesn't already exist, and the current directory location will be changed to it.

Some examples are:

```
asetup 15.6.5 --testarea <location> --multi --projtest
echo $TestArea
<location>/AtlasOffline-15.6.5
asetup 15.6.5  --testarea <location> --single
echo $TestArea
<location>
asetup 15.6.9.3 --testarea <location>  --multi --testtype="<base>"
echo $TestArea
<location>/15.6.9
```

Where:

- `<location>` is a directory location of your choice (e.g. `${HOME}/mytests`).

6. Testing of new AtlasSetup on the Grid.

Jaroslava Schovancova provided document on the whole ALRBdevel testing see https://hcdocs.web.cern.ch/hcdocs/atlas-ops/ALRBdevel.html

MRs for updates welcomeat : https://gitlab.cern.ch/hammercloud/hammercloud-docs/blob/master/atlas-ops/ALRBdevel.md

7. Platforms and Compilers

AtlasSetup has support for the following platforms and compilers. The compiler version shown in **bold** is the default for the corresponding platform, and will be selected by default if it's not explicitly specified. Similarly, the first numbered compiler version will be setup if the compiler version number is not explicitly specified. Thus specifying `gcc` will setup `gcc47` by default on SLC6.

- Linux: SLC6
  - Compilers: **gcc47**, gccmax, gcc48, gcc46, gcc43, icc13, clang32, clang33
- Linux: SLC5
  - Compilers: **gcc43**, gcc34, icc13, llvm28, lllvm27, llvm26
- Linux: SLC4
  - Compilers: **gcc34**
- MacOSX: 10.8 (Mountain Lion)
  - Compilers: **gcc46**
- MacOSX: 10.7 (Lion)
  - Compilers: **gcc42**
- MacOSX: 10.6 (Snow Leopard)
  - Compilers: **gcc42**, gcc40
- MacOSX: 10.5 (Leopard)
  - Compilers: **gcc40**

8. Environment variables setup by AtlasSetup

8.1 Runtime and CMT-related environment variables

The following Unix runtime and CMT-related environment variables are setup or modified by the `asetup` command:

```
CMTCONFIG
CMTEXTRATAGS
CMTHOME
CMTPATH
CMTPROJECTPATH
CMTSITE
DYLD_LIBRARY_PATH
LD_LIBRARY_PATH
MANPATH
PATH
SITEROOT
SVNROOT
SVNUSR
```

8.2 User and release related environment variables

In addition to the runtime and CMT-related environment variables listed in Section 7.1, asetup sets up two sets of environment variables:

- The user may define some environment variables in the `[environment]` section of configuration files as described in Section 7 of the AtlasSetup Reference Manual. Note that by default such environment variables are defined before the release itself it setup. This ensures that the user doesn't inadvertently override any variables that are critical to the correct functioning of the release (e.g. `PATH`). However, the experienced user can override this default prioritisation by use of the userpriority configuration variable. If this variable is set true, then the user specified environment variables will override those that are setup by the release. Great care should be taken to ensure that no unexpected interference occurs. Note also that the `[prolog.sh]` and `[epilog.sh]` (and the csh-equivalents) sections of configuration files can also be used to setup environment variables.
- Several release-related environment variables are setup to represent the current release and it;s runtime environment. These are detailed in AtlasSetup Reference Manual Section 12.

These are described in the AtlasSetup Reference Manual.

9. Pitfalls and differences relative to the use of AtlasLogin

- By default, AtlasSetup uses a search path to attempt to locate releases first in the kit installed `releases` area at CERN, and then in the directly AFS built `builds` area. This means that in general numbered releases (e.g. `15.6.9`) will be located in the `releases` area, and nightlies (e.g. `dev/rel_0`) will be located in the AFS `builds` area. This default may be modified by specifying the releases or standalone option to restrict the search to just the `releases` area, or by specifying the builds option to restrict the search to the =builds=area. The AtlasLogin default can be recovered by specifying the builds command line argument, or by adding the equivalent override to the users home configuration file.

- The AtlasSetup default configuration is for no test area to be setup whereas the default for AtlasLogin was to have a test area per release. The old default may be restored by using the multi or multitest argument or tag.

- By default, the AtlasLogin test area per release was named `<project>-<release>` (e.g. `AtlasOffline-15.6.9`), and the `simpletest` tag changed this default to be just `<release>` (e.g. `15.6.9`). The AtlasSetup default is for the test area per release to be named `<release>` (e.g. `15.6.9`). Use the projtest argument or tag to restore the default AtlasLogin behaviour.

- Since multi-character options must be prefixed by two dashes (e.g. `--tags`), the `-tag=` syntax used by the AtlasLogin package isn't supported, but must be replaced by the `--tags=` syntax (or just a list of arguments).

- By default, the version of CMT that was used to build a release is setup rather than it being under control of the user or developer. This default can be overridden using the cmtversion=<version> option/value pair.

- By default the full runtime for the release will be setup. If you wish to recover the `AtlasLogin` default and only setup a minimal build-time environment, use the nosetup or noruntime argument or tag.

- When using the runtime or setup argument or tag, the version of CMT that was used to build the release will be used, even if the final CMT version that is setup differs from this.

Major updates:
-- DavidQuarrie - 14-Aug-2013 -- ShuweiYe - 13-Mar-2018

Responsible: DavidQuarrie
Last reviewed by:

This topic: AtlasComputing > AtlasSetup
Topic revision: r137 - 2019-01-23 - EmilObreshkov