NAME

Lintian::Collect::Source – Lintian interface to source package data collection

SYNOPSIS

```
my ($name, $type, $dir) = ('foobar', 'source', '/path/to/lab-entry');
my $collect = Lintian::Collect->new ($name, $type, $dir);
if ($collect->native) {
    print "Package is native\n";
}
```

DESCRIPTION

Lintian::Collect::Source provides an interface to package data for source packages. It implements data collection methods specific to source packages.

This module is in its infancy. Most of Lintian still reads all data from files in the laboratory whenever that data is needed and generates that data via collect scripts. The goal is to eventually access all data about source packages via this module so that the module can cache data where appropriate and possibly retire collect scripts in favor of caching that data in memory.

CLASS METHODS

new (PACKAGE)

Creates a new Lintian::Collect::Source object. Currently, PACKAGE is ignored. Normally, this method should not be called directly, only via the Lintian::Collect constructor.

INSTANCE METHODS

In addition to the instance methods listed below, all instance methods documented in the Lintian::Collect and Lintian::Collect::Package modules are also available.

changelog

Returns the changelog of the source package as a Parse::DebianChangelog object, or `undef` if the changelog cannot be resolved safely.

Needs-Info requirements for using *changelog*: Same as index_resolved_path

diffstat

Returns the path to diffstat output run on the Debian packaging diff (a.k.a. the ‘‘diff.gz’’) for 1.0 non-native packages. For source packages without a ‘‘diff.gz’’ component, this returns the path to an empty file (this may be a device like /dev/null).

Needs-Info requirements for using *diffstat*: diffstat

native

Returns true if the source package is native and false otherwise. This is generally determined from the source format, though in the 1.0 case the nativeness is determined by looking for the diff.gz (using the name of the source package and its version).

If the source format is 1.0 and the version number is absent, this will return false (as native packages are a lot rarer than non-native ones).

Note if the source format is missing, it is assumed to be a 1.0 package.

Needs-Info requirements for using *native*: Same as field

version

Returns a fully parsed Lintian::Info::Changelog::Version for the source package’s version string.

Needs-Info requirements for using *version*: Same as field

files

Returns a reference to a hash containing information about files listed in the .changes file. Each hash may have the following keys:

name
: Name of the file.

size
: The size of the file in bytes.

checksums
: A hash with the keys being checksum algorithms and the values themselves being hashes containing

  sum
  : The result of applying the given algorithm to the file.

  filesize
  : The size of the file as given in the .changes section relating to the given checksum.

Needs-Info requirements for using *files*: ''field ([FIELD[, DEFAULT]])'' in Lintian::Collect

repacked
: Returns true if the source package has been ''repacked'' and false otherwise. This is determined from the version name containing ''dfsg'' or similar.

  Needs-Info requirements for using *repacked*: Same as field

binaries
: Returns a list of the binary and udeb packages listed in the *debian/control*. Package names appear the same order in the returned list as they do in the control file.

  Note: Package names that are not valid are silently ignored.

  Needs-Info requirements for using *binaries*: Same as binary_package_type

binary_package_type (BINARY)
: Returns package type based on value of the Package-Type (or if absent, X−Package-Type) field. If the field is omitted, the default value ''deb'' is used.

  If the BINARY is not a binary listed in the source packages *debian/control* file, this method return `undef`.

  Needs-Info requirements for using *binary_package_type*: Same as binary_field

source_field([FIELD[, DEFAULT]])
: Returns the content of the field FIELD from source package paragraph of the *debian/control* file, or DEFAULT (defaulting to `undef`) if the field is not present. Only the literal value of the field is returned.

  If FIELD is not given, return a hashref mapping field names to their values (in this case DEFAULT is ignored). This hashref should not be modified.

  NB: If a field from the ''dsc'' file itself is desired, please use field instead.

  Needs-Info requirements for using *source_field*: Same as index_resolved_path

orig_index (FILE)
: Like ''index'' except orig_index is based on the ''orig tarballs'' of the source packages.

  For native packages ''index'' and ''orig_index'' are generally identical.

  NB: If sorted_index includes a debian packaging, it is was contained in upstream part of the source package (or the package is native).

  Needs-Info requirements for using *orig_index*: src-orig-index

sorted_orig_index
: Like sorted_index except sorted_orig_index is based on the ''orig tarballs'' of the source packages.

  For native packages sorted_index and ''sorted_orig_index'' are generally identical.

NB: If sorted_orig_index includes a debian packaging, it is was contained in upstream part of the source package (or the package is native).

Needs-Info requirements for using *sorted_orig_index*: Same as orig_index

orig_index_resolved_path(PATH)
Resolve PATH (relative to the root of the package) and return the entry denoting the resolved path.

The resolution is done using resolve_path.

NB: If orig_index_resolved_path includes a debian packaging, it is was contained in upstream part of the source package (or the package is native).

Needs-Info requirements for using *orig_index_resolved_path*: Same as orig_index

binary_field (PACKAGE[, FIELD[, DEFAULT]])
Returns the content of the field FIELD for the binary package PACKAGE in the *debian/control* file, or DEFAULT (defaulting to `undef`) if the field is not present. Inheritance of field values from the source section of the control file is not implemented. Only the literal value of the field is returned.

If FIELD is not given, return a hashref mapping field names to their values (in this case, DEFAULT is ignored). This hashref should not be modified.

If PACKAGE is not a binary built from this source, this returns DEFAULT.

Needs-Info requirements for using *binary_field*: Same as index_resolved_path

java_info
Returns a hashref containing information about JAR files found in source packages, in the form *file name* -> *info*, where *info* is a hash containing the following keys:

manifest
A hash containing the contents of the JAR file manifest. For instance, to find the classpath of *`$file`*, you could use:

```
if (exists $info->java_info->{$file}{'manifest'}) {
    my $cp = $info->java_info->{$file}{'manifest'}{'Class-Path'};
    # ...
}
```

NB: Not all jar files have a manifest. For those without, this will value will not be available. Use exists (rather than defined) to check for it.

files
A table of the files in the JAR. Each key is a file name and its value is its ''Major class version'' for Java or ''-'' if it is not a class file.

error
If it exists, this is an error that occurred during reading of the zip file. If it exists, it is unlikely that the other fields will be present.

Needs-Info requirements for using *java_info*: java-info

binary_relation (PACKAGE, FIELD)
Returns a Lintian::Relation object for the specified FIELD in the binary package PACKAGE in the *debian/control* file. FIELD should be one of the possible relationship fields of a Debian package or one of the following special values:

all The concatenation of Pre-Depends, Depends, Recommends, and Suggests.

strong
The concatenation of Pre-Depends and Depends.

weak
The concatenation of Recommends and Suggests.

If FIELD isn't present in the package, the returned Lintian::Relation object will be empty (always satisfied and implies nothing).

Any substvars in *debian/control* will be represented in the returned relation as packages named after the substvar.

Needs-Info requirements for using *binary_relation*: Same as binary_field

relation (FIELD)
Returns a Lintian::Relation object for the given build relationship field FIELD. In addition to the normal build relationship fields, the following special field names are supported:

build-depends-all
The concatenation of Build-Depends, Build-Depends-Arch and Build-Depends-Indep.

build-conflicts-all
The concatenation of Build-Conflicts, Build-Conflicts-Arch and Build-Conflicts-Indep.

If FIELD isn't present in the package, the returned Lintian::Relation object will be empty (always satisfied and implies nothing).

Needs-Info requirements for using *relation*: Same as field

relation_noarch (FIELD)
The same as ''relation (FIELD)'', but ignores architecture restrictions and build profile restrictions in the FIELD field.

Needs-Info requirements for using *relation_noarch*: Same as relation

debfiles ([FILE])
This method is deprecated. Consider using index_resolved_path(PATH) instead, which returns Lintian::Path objects.

Returns the path to FILE in the debian dir of the extracted source package. FILE must be relative to the root of the debian dir and should be without leading slash (and without ''./''). If FILE is not in the debian dir, it returns the path to a non-existent file entry.

It is not permitted for FILE to be `undef`. If the ''root'' dir is desired either invoke this method without any arguments at all or use the empty string.

The caveats of unpacked also apply to this method.

Needs-Info requirements for using *debfiles*: debfiles

index (FILE)
For the general documentation of this method, please refer to the documentation of it in Lintian::Collect::Package.

The index of a source package is not very well defined for non-native source packages. This method gives the index of the ''unpacked'' package (with 3.0 (quilt), this implies patches have been applied).

If you want the index of what is listed in the upstream orig tarballs, then there is ''orig_index''.

For native packages, the two indices are generally the same as they only have one tarball and their debian packaging is included in that tarball.

IMPLEMENTATION DETAIL/CAVEAT: Lintian currently (2.5.11) generates this by running ''**find** (1)'' after unpacking the source package. This has three consequences.

First it means that (original) owner/group data is lost; Lintian inserts ''root/root'' here. This is usually not a problem as owner/group information for source packages do not really follow any standards.

Secondly, permissions are modified by A) umask and B) laboratory set{g,u}id bits (the laboratory on

lintian.d.o has setgid). This is *not* corrected/altered. Note Lintian (usually) breaks if any of the ''user'' bits are set in the umask, so that part of the permission bit *should* be reliable.

Again, this shouldn't be a problem as permissions in source packages are usually not important. Though if accuracy is needed here, ''orig_index'' may used instead (assuming it has the file in question).

Third, hardlinking information is lost and no attempt has been made to restore it.

Needs-Info requirements for using *index*: unpacked

is_non_free
Returns a truth value if the package appears to be non-free (based on the section field; ''non−free/*'' and ''restricted/*'')

Needs-Info requirements for using *is_non_free*: ''source_field ([FIELD[, DEFAULT]])''

AUTHOR

Originally written by Russ Allbery <rra@debian.org> for Lintian.

SEE ALSO

lintian (1), **Lintian::Collect** (3), **Lintian::Relation** (3)