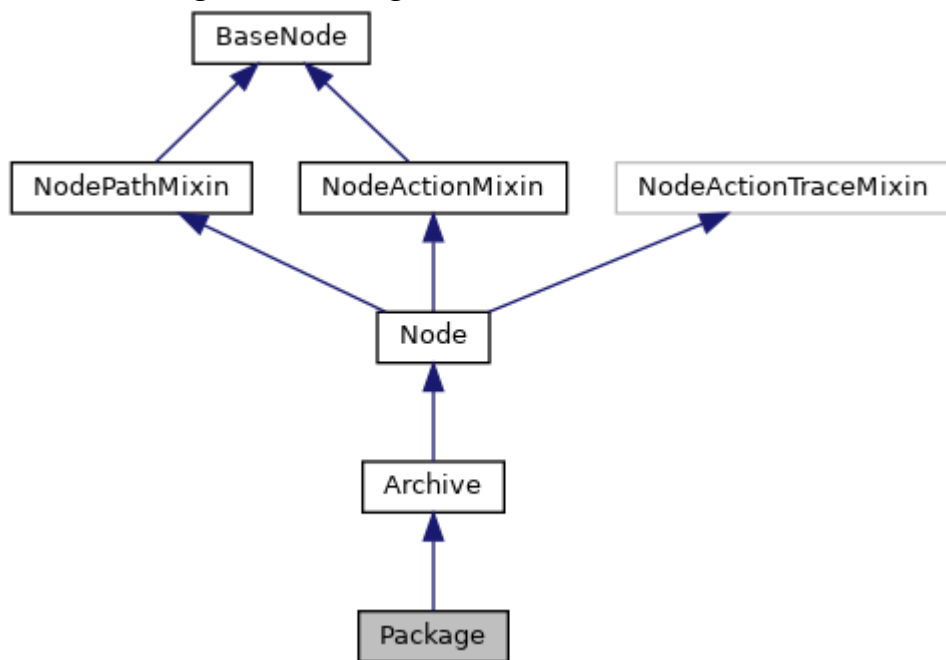


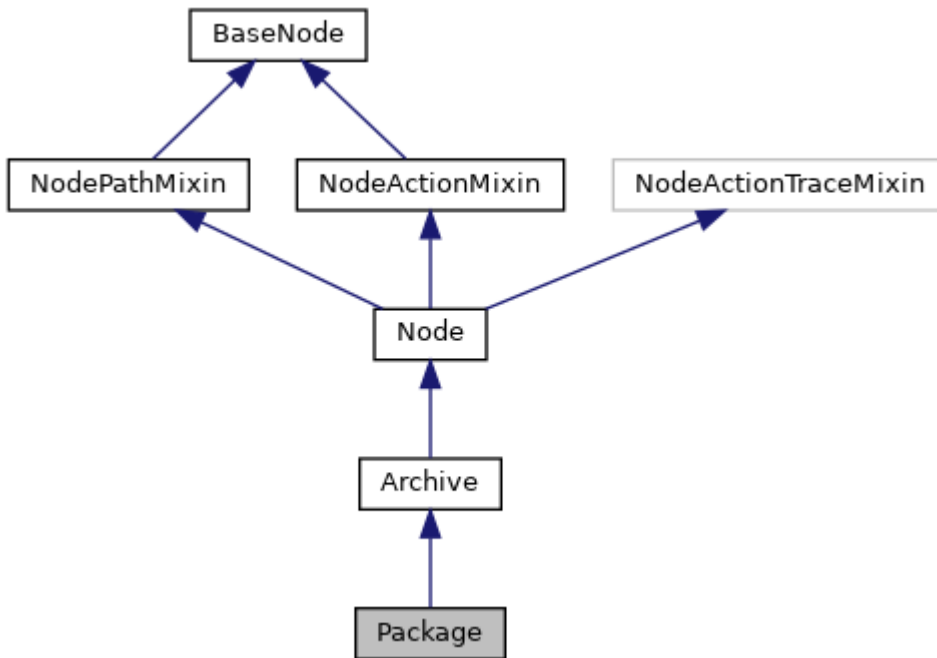
Package Class Reference

Inheritance diagram for Package:



[\[legend\]](#)

Collaboration diagram for Package:



[\[legend\]](#)

Public Member Functions

def [__init__](#) (self, [platform](#))

def [new_archive_child](#) (self)

► Public Member Functions inherited from [Archive](#)

def [control_node](#) (self)

def [control](#) (self)

def [set_signer](#) (self, signer_type)

► Public Member Functions inherited from [Node](#)

def [__init__](#) (self, [name](#)=None, [parent](#)=None, [tarinfo](#)=None)

► Public Member Functions inherited from [NodePathMixin](#)

def [get_path_nodes](#) (self, include_root=False, within_archive=False)

str [get_path](#) (self, include_root=False, within_archive=False)

Retrieve path of this node within the archive or withing whole tree of archives Separator '/' is used within archives and for recursion into archives. [More...](#)

int [get_depth](#) (self, within_archive=False)

Retrieve depth of node within the archive or within whole tree of archives. [More...](#)

def [find](#) (self, str node_path)

Find node according to node path provided. [More...](#)

def [find_re](#) (self, str node_path_re)

Find node according to regular expression node path provided Search will look for regex matching node name match along the path. [More...](#)

bool [has_ancestor](#) (self, ancestor)

Find if node has another node as ancestor. [More...](#)

► Public Member Functions inherited from [BaseNode](#)

str [get_type_str](#) (self)
Returns node type as string Supported are: "Package", "Bundle", "Dlfile", "Signature", "Control", "Manifest", "Dir", "File", "Symlink". [More...](#)

bool [is_csd](#) (self)
Test if node is a CSD. [More...](#)

bool [is_inf](#) (self)
Test if node is an INF. [More...](#)

bool [is_manifest](#) (self)
Test if node is a manifest. [More...](#)

bool [is_changeset](#) (self)
Test if node is a changeset. [More...](#)

bool [is_archive](#) (self)
Test if node is an archive. [More...](#)

bool [is_dlfile](#) (self)
Test if node is a dlfile. [More...](#)

bool [is_bundle](#) (self)
Test if node is a bundle. [More...](#)

bool [is_package](#) (self)
Test if node is a package. [More...](#)

bool [is_signature](#) (self)
Test if node is a signature. [More...](#)

bool [is_control](#) (self)
Test if node is a control. [More...](#)

def [mark_modified](#) (self)

def [get_parent_archive](#) (self)
Retrieve parent archive node. [More...](#)

def [get_platform](#) (self)
Retrieve platform for this node. [More...](#)

def [get_root_archive](#) (self)
Retrieve root archive node Root archive node has no parent archive. [More...](#)

int [get_index](#) (self)
Retrieve index of this node in the list of its parent's children. [More...](#)

def [get_signer](#) (self)

def [__str__](#) (self)

► Public Member Functions inherited from [NodeActionMixin](#)

def [unlink](#) (self)
Remove this node from tree. [More...](#)

def [destroy](#) (self)
Remove and destroy this node and all its children. [More...](#)

def [add_child](#) (self, child_node, index=None)
Add a child node. [More...](#)

def [create_dir](#) (self, dir_name)
Add a child directory If same child name exists, it will be replaced. [More...](#)

def [create_file](#) (self, file_name, content=b")
Add a child file. [More...](#)

def [set_content](#) (self, content)
Set the content of this file node. [More...](#)

def [set_mode](#) (self, mode)
Set the mode of this file/directory. [More...](#)

def [copy](#) (self)
Make a deep copy of this node and its content, children and attributes. [More...](#)

def [move](#) (self, [parent](#))
Move this node to another parent. [More...](#)

def [rename](#) (self, newname)
Rename this node and underlying tarinfo name. [More...](#)

Static Public Member Functions

def [create_package](#) ([platform](#), base_name, stem_name=None, control_content=None)
Create a package node It also creates a CONTROL directory and a control file in it. [More...](#)

► Static Public Member Functions inherited from [Archive](#)

def [setup_archive](#) (archive, full_name)
► Static Public Member Functions inherited from [Node](#)

def [create_node](#) ([name](#), entry_type)
► Static Public Member Functions inherited from [BaseNode](#)

tarfile.TarInfo [create_tarinfo](#) (str tarinfo_name, entry_type)
Tarinfo creation helper. [More...](#)

Additional Inherited Members

► Data Fields inherited from [Archive](#)

[platform](#)
[tar](#)
[is_gzip](#)
[pax_headers_found](#)
[signer](#)

► Data Fields inherited from [BaseNode](#)

[name](#)
node name (str) [More...](#)

[parent](#)

parent node ([BaseNode](#)) [More...](#)

[children](#)

children of this node in order. [More...](#)

[tarinfo](#)

Tarinfo object containing the type of archive entry, path, user, group, permission, etc.

[More...](#)

[file_content](#)

Object representing the content of the file. [More...](#)

[signature_nodes](#)

Nodes holding signature of this node. [More...](#)

► Data Fields inherited from [NodeActionMixin](#)

[parent](#)

[children](#)

[file_content](#)

[name](#)

[signature_nodes](#)

Detailed Description

Represents a package node

Constructor & Destructor Documentation

[?__init__\(\)](#)

```
def __init__( self,  
              platform  
            )
```

Reimplemented from [Archive](#).

Member Function Documentation

[?create_package\(\)](#)

```
def create_package ( platform,
                    base_name,
                    stem_name = None,      static
                    control_content = None
                    )
```

Create a package node It also creates a CONTROL directory and a control file in it.

Name of node will be '[stem_name].tgz' or '[stem_name].tar' depending on platform

Parameters

platform	platform name or instance
base_name	name to be used in control file
stem_name	will be 'pkg.[base_name]' if not provided
control_content	bytes content of the control file. Name and version are populated if not provided.

Returns

this node

[?](#) **new_archive_child()**

```
def new_archive_child ( self )
```

The documentation for this class was generated from the following file:

- sys/import/packman/packmanlib/[node.py](#)