

## Contents

1	NAME	2
2	DESCRIPTION	2
3	MORE INFO	2
4	PLUGIN API	2

## 1 NAME

OpenSLX::OSPlugin::Base - the base class for all OpenSLX OS-plugins.

## 2 DESCRIPTION

This class defines the OpenSLX API for OS-plugins.

The general idea behind OS-plugins is to extend any installed vendor-OS with a specific features. Each feature is implemented as a separate, small software component in order to make them easy to understand and maintain.

Since all of these software components are plugged into the OpenSLX system by means of a common API, we call them **OS-plugins**.

This API can be separated into different parts:

- **Declarative Interface** (provide info about a plugin)
- **Vendor-OS Interface** (installing or removing a plugin into/from a vendor-OS)
- **Initramfs Interface** (integrating a plugin into an initramfs)

## 3 MORE INFO

Please read the user-level introduction on plugins in the OpenSLX-wiki: <http://openslx.org/trac/de/openslx/> (in German).

If you'd like to know how a plugin is implemented, please have a look at the 'example' plugin, which contains some explanations and useful hints.

If you have any questions regarding the concept of OS-plugins and their implementation, please drop a mail to: [ot@openslx.com](mailto:ot@openslx.com), or join the IRC-channel '#openslx' (on freenode).

## 4 PLUGIN API

### Declarative Interface

#### **new()**

Every plugin should provide a new-method and provide it's own name in the 'name' entry of \$self.

Please note that by convention, plugin names are all lowercase!

#### **initialize()**

Initializes basic context for this plugin (esp. a reference to the OSPlugin engine that drives this plugin).

### **getInfo()**

Returns a hash-ref with administrative information about this plugin (what does it do and how does it relate to other plugins). Every plugin needs to provide this method and return the information about itself.

The returned hash-ref must include at least the following entries:

#### **description**

Explains the purpose of this plugins.

#### **precedence**

Specifies the execution precedence of this plugin with respect to all other plugins (plugins with lower precedences will be started before the ones with a higher precedence).

Valid values range from 0-99. If your plugin does not have any requirements in this context, just specify the default value '50'.

#### **required**

Specifies the list of plugins that are required by this plugin.

Before any plugin can be installed, all other plugins that are required by it must have been installed.

### **getAttrInfo()**

Returns a hash-ref with information about all attributes supported by this specific plugin.

This default configuration will be added as attributes to the default system, such that it can be overruled for any specific system by means of **slxconfig**.

The returned hash-ref must include at least the following entries:

#### ***plugin-name::active***

Indicates whether or not this plugin is active (1 for active, 0 for inactive).

### **getDefaultAttrsForVendorOS()**

Returns a hash-ref with the default attribute values for the given vendor-OS.

### **checkStage3AttrValues()**

Checks if the stage3 values given in **\$stage3Attrs** are allowed and make sense.

This method returns an array-ref of problems found. If there were no problems, this methods returns undef.

Plugins may override this implementation to do checks that for instance look at the stage1 vendor-OS-attributes given in **\$vendorOSAttrs**.

N.B.: this method is called while being chrooted into the vendor-OS, so it may invoke all distro methods that expect to be run in this environment, too

**dependsOnPlugin()**

## Vendor-OS Interface

**installationPhase()**

In this method, the plugin should install itself into the given vendor-OS.

What "installation" means is up to the plugin. Some plugins may just copy a file from the OpenSLX host installation into the vendor-OS, while others may need to download files from the internet and/or install packages through the vendor-OS' meta packager.

N.B.: This method is invoked while chrooted into the vendor-OS root.

The hash-ref given in **\$info** contains vital information for the installation process:

**plugin-repo-path**

The folder where the stage1-plugin should store all files required by the corresponding stage3 runlevel script.

**plugin-temp-path**

A temporary playground that will be cleaned up automatically.

**openslx-base-path**

In order to make the OpenSLX files from the host available, the OpenSLX base folder (normally /opt/openslx) will be mounted into the chroot. So if you have to copy any files from the host, fetch them from this path.

**openslx-config-path**

In order to make the OpenSLX config files from the host available, the OpenSLX config folder (normally /etc/opt/openslx) will be mounted into the chroot. So if you have to copy any config files from the host, fetch them from this path.

**plugin-attrs**

Contains the attributes in effect for the installation of this plugin.

**removalPhase()**

In this method, the plugin should remove itself from the given vendor-OS.

What "removal" means is up to the plugin. Some plugins may just delete a file from the vendor-OS, while others may need to uninstall packages through the vendor-OS' meta packager.

N.B.: This method is invoked while chrooted into the vendor-OS root.

The hash-ref given in **Sinfo** contains vital information for the installation process:

**plugin-repo-path**

The folder where the stage1-plugin should store all files required by the corresponding stage3 runlevel script.

**plugin-temp-path**

A temporary playground that will be cleaned up automatically.

**openslx-base-path**

In order to make the OpenSLX files from the host available, the OpenSLX base folder (normally /opt/openslx) will be mounted into the chroot. So if you have to copy any files from the host, fetch them from this path.

**openslx-config-path**

In order to make the OpenSLX config files from the host available, the OpenSLX config folder (normally /etc/opt/openslx) will be mounted into the chroot. So if you have to copy any config files from the host, fetch them from this path.

**plugin-attrs**

Contains the attributes in effect for the installation of this plugin.

**preInstallationPhase()**

In this method, any preparations for installation of the plugin into a vendor-OS should be executed. As this method is being called immediately before the chroot is entered, this is the last/only chance to copy any files into the chroot that are required from within (in installationPhase()).

The given parameters are similar to the ones for installationPhase(), except that all paths are now relative to the root-fs instead of being relative to the chroot (i.e. the paths are ready to be used from outside the chroot):

A "exit 1;" will result in a not installed plugin.

**plugin-repo-path**

The folder where the stage1-plugin should store all files required by the corresponding stage3 runlevel script.

**plugin-temp-path**

A temporary playground that will be cleaned up automatically.

If a plugin needs to unpack any archives, these archives should be copied to this folder (as it will be cleaned automatically).

**openslx-base-path**

In order to make the OpenSLX files from the host available, the OpenSLX base folder (normally /opt/openslx) will be mounted into the chroot. So if you have to copy any files from the host, fetch them from this path.

**openslx-config-path**

In order to make the OpenSLX config files from the host available, the OpenSLX config folder (normally `/etc/opt/openslx`) will be mounted into the chroot. So if you have to copy any config files from the host, fetch them from this path.

**plugin-attrs**

Contains the attributes in effect for the installation of this plugin.

**vendor-os-path**

Contains the path to the vendor-OS into which the plugin will be installed.

**postRemovalPhase()**

In this method, any plugin has the chance to do any necessary cleanup that must be executed outside of the chroot.

This method is invoked immediately after leaving the chroot into the vendor-OS root, but before the `plugin-temp-path` has been cleaned up. So if required, any files could be copied out of the `temp-path` somewhere into the `root-fs`.

The given parameters are similar to the ones for `removalPhase()`, except that all paths are now relative to the `root-fs` instead of being relative to the chroot (i.e. the paths are ready to be used from outside the chroot):

**plugin-repo-path**

The folder where the `stage1-plugin` should store all files required by the corresponding `stage3` runlevel script.

**plugin-temp-path**

A temporary playground that will be cleaned up automatically.

**openslx-base-path**

In order to make the OpenSLX files from the host available, the OpenSLX base folder (normally `/opt/openslx`) will be mounted into the chroot. So if you have to copy any files from the host, fetch them from this path.

**openslx-config-path**

In order to make the OpenSLX config files from the host available, the OpenSLX config folder (normally `/etc/opt/openslx`) will be mounted into the chroot. So if you have to copy any config files from the host, fetch them from this path.

**plugin-attrs**

Contains the attributes in effect for the installation of this plugin.

**vendor-os-path**

Contains the path to the vendor-OS from which the plugin has been removed.

## Initramfs Interface

All of the following methods are invoked by the config demuxer when it makes an initramfs for a system that has this plugin activated. Through these methods, each plugin can integrate itself into that initramfs.

### **suggestAdditionalKernelParams()**

Called in order to give the plugin a chance to add any kernel params it requires.

In order to do so, the plugin should return a list of additional kernel params that it would like to see added.

### **suggestAdditionalKernelModules()**

Called in order to give the plugin a chance to add any kernel modules it requires.

In order to do so, the plugin should return the names of additional kernel modules that it would like to see added.

### **copyRequiredFilesIntoInitramfs()**

Called in order to give the plugin a chance to copy all required files from the vendor-OS into the initramfs.

N.B.: Only files that are indeed required by the initramfs should be copied here, i.e. files that are needed *\*before\** the root-fs has been mounted. All other files should be taken from the root-fs instead!

### **setupPluginInInitramfs()**

Called in order to let the plugin setup all the files it requires in the initramfs.

Normally, you don't need to override this method in your own plugin, as it is usually enough to override `suggestAdditionalKernelParams()`, `suggestAdditionalKernelModules()` and maybe `copyRequiredFilesIntoInitramfs()`.

1;