# Directory Based C++ Plugin Configuration

Unknown macro: 'xxfuture'

**This page describes how the configuration of C++ plugins is changing in 4.3.**

For related issues, see #Tracking below.

## Configuration File Names and Structure

With one exception, the interface to a plugin, has not changed; the reading of configuration data in a plugin is backwards compatible with the implementation in version 4.2.

The exception is a #Redirect Plugin Interface Change

There are 4 parts to the configuration for a plugin (with the colors used for them below):

- The hook that the plugin module should be called by (effectively what subclass of 'Plugin' it is); shown in red.
- The name of the plugin instance (it is possible to configure more than one copy of the same plugin with different names, each with its configuration values); shown in blue
- The shared library that implements the plugin; shown in green
- Any configuration name/value pairs required by the plugin; shown in orange

### The Old Way

In 4.2 and earlier, all plugins for a component were configured by adding lines to the (single) `*-config` file for the component; in the case of the registrar, this is `/etc/sipxpbx/registrar-config`

Every plugin configuration required a line to configure the library, but parts of that line also provided the hook and the instance. This configuration line was constructed as:

```
hookname_HOOK_LIBRARY.instancename : librarypath
```

Each line in the configuration that matched that pattern configured the existence of a plugin instance and specified its library. The line in the `registrar-config` file that adds the mapping rules redirector plugin is:

```
SIP_REDIRECT_HOOK_LIBRARY.130-MAPPING : /usr/lib/libRedirectorMapping.so
```

If no configuration parameters are needed by the plugin, that that one line is all that was needed. When the plugin needs parameters, the configuration lines for those values were constructed as:

```
hookname.instancename.paramname : paramvalue
```

The directive that configures the file name for the mapping rules file name is:

```
SIP_REDIRECT.130-MAPPING.MAPPING_RULES_FILENAME : /etc/sipxpbx/mappingrules.xml
```

### The New Way

In 4.3 each hook for a component has a directory into which a separate configuration file is written for each plugin instance. So, for the registrar, which has two different plugin hooks, there will be two directories:

```
/etc/sipxpbx/redirect-hook/
/etc/sipxpbx/registrar-hook/
```

The individual plugin configuration file name is contructed using the instance name and the suffix `.plugin`:

```
instancename.plugin
```

The contents of the plugin configuration files are simpler in the new scheme, since encoding the hook and instance names in the contents is not required.

To configure the library name every `.plugin` configuration file must contain at least a `HOOK_LIBRARY` value:

```
HOOK_LIBRARY : librarypath
```

If no configuration parameters are needed by the plugin, then that one line is all that is needed.

When the plugin does need parameters, the configuration lines for those values are just the parameter name and value:

```
paramname : paramvalue
```

So, to add the mappingrules redirector plugin and assign its file name, the file `/etc/sipxpbx/redirect-hook/130-mapping.plugin` is written:

```
HOOK_LIBRARY : /usr/lib/libRedirectorMapping.so
MAPPING_RULES_FILENAME : /etc/sipxpbx/mappingrules.xml
```

## List of Plugins

| Component | PluginClass | Description | Old Prefix | New Directory |
|---|---|---|---|---|
| sipXregistry | RegisterPlugin | Registration side effects | SIP_REGISTRAR | /etc/sipxpbx/registrar-hook |
| sipXregistry | RedirectPlugin | Routing lookups | SIP_REDIRECT | /etc/sipxpbx/redirect-hook |
| sipXproxy | AuthPlugin | Authorization checks | SIPX_PROXY | /etc/sipxpbx/auth-hook |

## Redirect Plugin Interface Change

If you have written a `RedirectPlugin` that is not in the sipXecs sources, then this interface change is important to you; if not, you can ignore it.

In 4.2, the authority level of a plugin was configured using the naming convention of a plugin parameter, like this:

```
SIP_REDIRECT.130-MAPPING.AUTHORITY_LEVEL : 40
```

but it was actually read and acted on by the SipRouter class in the redirect server directly. This was cumbersome, but reasonable since all the directives were in the same configuration file.

In order to avoid the SipRouter needing to scan the now-separate plugin configurations a second time (the first being the scan that instatiates the plugins), the base RedirectPlugin class has had two methods added:

- **`void readAuthorityLevel(OsConfigDb& configDb`** which must now be called from the `readConfig` method of any plugin to read the authority level configuration from its own configuration file
- **`int authorityLevel(void)`** which is now called from the SipRouter after the configurations are loaded. The default base class implementation of `readConfig` has been modified to make this call.

## Tracking

| type | key | summary | assignee | reporter | priority | status | resolution | created | updated | due |
|---|---|---|---|---|---|---|---|---|---|---|

Unable to locate Jira server for this macro. It may be due to Application Link configuration.