How to build an e-smith contrib module

Kirrily Robert
e-smith, inc.

1. Introduction

The e-smith server and gateway is a modular Linux system permitting the installation of add-on/contributed software. This software is available primarily from the e-smith developer website (http://www.e-smith.org/) in the form of RPM packages.

This document describes how a 3rd party developer can create an RPM package suitable for use with the e-smith server and gateway. It is aimed at those who are:

- moderately familiar with the e-smith server and gateway system (i.e. have installed and used it, and read the user guide)
- have some familiarity with the Linux/UNIX command line
- have some proficiency in programming in Perl, or are able to quickly understand new programming languages

1.1. Version information

The CVS version of this document is:

`$Id: building-contribs.sgml,v 1.14 2001/06/11 14:07:17 dyork Exp $`

It applies to e-smith version 4.1.

1.2. Feedback, comments, etc.

If you have any comments or feedback on this document, please email <`documentation@e-smith.com`>.

1.3. Where to find e-smith documentation

This and other HOWTOs, papers, and the full e-smith manual can be found on the e-smith developer website (http://www.e-smith.org/docs/)

2. Components of an e-smith add-on module

An e-smith add-on module typically consists of the following parts:

- RPM infrastructure
- Underlying software
- Templated configuration files

- Web-based manager panel(s)
- Additional scripts, files, etc.

The following sections will describe each of these in detail.

3. RPM infrastructure

3.1. A quick introduction to RPMs

All e-smith add-on modules are distributed as RPM packages. This is the format used by Red Hat and various other Linux distributions for distributing applications and other collections of files.

An RPM essentially consists of a tarball (a compressed archive) of all the files required by a piece of software. Additionally, it includes meta-information describing the software, and scripts which must be run to install or uninstall the software.

Meta-information stored in an RPM includes:

1. summary and description of the software
2. package name
3. version number
4. copyright information
5. category/group to which the software belongs
6. name and email address of the packager
7. pre-requisites to installing this package
8. ... and more

3.2. Setting up your RPM environment

To create RPMs efficiently, you need to set up a space in which to do your work. You will probably want to do this on a separate system to your live e-smith server and gateway, as the e-smith system doesn't come with all the tools required for development (compilers, etc.) and installing them could constitute a security risk. Two possible options are a Red Hat system with similar versions of software to the e-smith server and gateway (for example Red Hat Linux 7.0 for e-smith 4.1), or a separate "development" e-smith server with appropriate RPMs installed for development work (for example a compiler such as **gcc**).

Here are the steps used to set up an RPM development environment:

```
# make yourself an RPM playpen - doing things this way means you don't
# have to be root to build things.
mkdir -p rpms/{SRPMS,BUILD,SOURCES,SPECS,RPMS,lib}
mkdir -p rpms/RPMS/{i386,noarch}
rpm -initdb -dbpath ~/rpms/lib
# configure rpm to use this playpen
echo "%_topdir $HOME/rpms" > ~/.rpmmacros
```

You will now find that you have a directory called `rpms` in which you will do your work. Under this are the following subdirectories:

`SOURCES`
: The base material from which RPMs are built – source code, tarballs, etc.

`BUILD`
: Working area used by the **rpm** program during RPM creation – do not touch

`SPECS`
: Specification files for building RPMs

`SRPMS`
: Source RPMS (created by build process)

`RPMS`
: Binary RPMS (created by build process). Has subdirectories `noarch` and `i386` for architecture independent and x86 platforms respectively.

As you prepare software to turn into RPMs, you will place files in these directories as appropriate. The following sections will describe what goes where as each item is covered.

> **Tip:** As you start work on an RPM for version x.y.z of a package, create a subdirectory `rpms/SOURCES/yourpackage-x.y.z/` to work in.
>
> ```
> mkdir rpms/SOURCES/yourpackage-x.y.z
> ```
>
> Under this directory there should be a subdirectory called `root`, under which is an image of the file hierarchy that will be installed by the RPM.
>
> ```
> mkdir rpms/SOURCES/yourpackage-x.y.z/root
> ```

4. Underlying software

e-smith contrib modules usually come in one of the following flavours:

1. software developed entirely for e-smith (example: tools to manipulate the configuration database)
2. existing software already available as an RPM (example: PostgreSQL)
3. existing software not already available as an RPM
4. existing software which needs to be shifted around considerably to work with e-smith (example: many web applications)

This section on underlying software does not apply to the first type, software developed entirely for e-smith. If you are writing purely e-smith software, skip to the next section.

4.1. Existing software available as an RPM

Usually this software can just be installed on an e-smith system with no modifications. However, such an installation will need extra bits such as the templated configuration files and manager interfaces to work properly.

Take, for example, an imaginary software package called CoolApp. There is an RPM called `CoolApp.rpm` which can be installed on an e-smith system. To make it work well on an e-smith system, we create a separate RPM called `e-smith-CoolApp.rpm` which contains the e-smith specific parts. This way we don't need to mess with the underlying software at all.

4.2. Existing software not available as an RPM

If the existing software is not available already as an RPM, you will need to make one. This document does not deal specifically with how to package RPMs of third-party software, but many web resources can help you with it. See http://www.rpm.org/RPM-HOWTO/ for starters.

Tip: Before you build your own RPM, do a search on rpmfind.net (http://www.rpmfind.net) – there's a *lot* of software packaged as RPMs.

4.3. Existing software which needs major modifications to work with e-smith

As mentioned above, the most common examples of such software are web applications. These often need to be installed by hand into an appropriate web-accessible directory, and may need other modifications to their functionality or their look and feel to work smoothly with e-smith.

To make an RPM of such software, you need to create an image of the installed software in your `rpms/SOURCES/` directory.

If the software you are packaging would install files in `/home/e-smith/files/ibays/webthing/` they should be placed in `rpms/SOURCES/yourpackage-x.y.z/root/home/e-smith/files/ibays/webthing/`.

5. Templated configuration files

The e-smith server and gateway uses a sophisticated template system to create configuration files. These are kept in `/etc/e-smith/templates/` on a live system, and hence should be created in `rpms/SOURCES/yourpackage-x.y.z/root/etc/e-smith/templates` in your RPM workspace.

For a detailed discussion of the e-smith templated configuration system and how to use it to greatest effect, see the paper The e-smith templated configuration system (http://www.e-smith.org/docs/papers/templates.html) on the e-smith developer website.

6. Creating manager panels

Most e-smith add-ons will require an interface to allow them to be administered via the web manager.

The web manager's navigation frame is generated automatically by examining the contents of the directory `/etc/e-smith/web/functions`. Each file in this directory is a Perl CGI script which generates the content of the main frame and deals with the results of the form being submitted. The easiest way to figure out how to write one is to modify an existing script to suit your needs.

If you are not already familiar with the Perl programming language, you will need to read up on at least the basics. One online course is available from http://sourceforge.net/projects/spork.

The e-smith server and gateway comes with certain Perl libraries to perform common functions including manipulating the configuration database, performing common CGI tasks, etc. The modules available include:

- `esmith::util`
- `esmith::db`
- `esmith::cgi`

The documentation can be accessed from the Linux command line on your e-smith server by typing **perldoc esmith::cgi** (or whatever module name you're interested in). The documentation is also available in HTML format at http://www.e-smith.org/docs/perldoc/.

> **Note:** `esmith::config` is an older module which provides access to the e-smith configuration database. It has been deprecated in favour of esmith::db which is a more abstracted version offering access to any file in the configuration file's format. Please use `esmith::db` in preference to `esmith::config`.

In order to be listed in the navigation panel, your Perl script must contain the following comment lines, usually at the top of the script:

```
# heading     : Configuration
# description : Email retrieval
# navigation  : 2000 2700
```

These define the category heading under which your add-on's admin interface should be listed, the title it should have, and the priority it should have in the listing order. The first number gives the priority of the heading (usually a multiple of 1000) and the second number gives the priority of this particular item within that heading group. In other words, a heading with a priority of 1000 will come before one with 2000 in the navigation panel, and within that heading category the individual items are listed in order from highest to lowest.

To figure out what numbers to give your own script, figure out where you want it to appear in the navigation panel then check source code for the scripts which appear before and after where you want to be. For instance, if you want your item to appear before "Remote Access" and after "Local Networks" in the navigation menu, you would look at `/etc/e-smith/web/functions/remoteaccess` and `/etc/e-smith/web/functions/localnetworks` and find the following:

```
# heading     : Security
# description : Remote access
# navigation  : 1000 1200
# heading     : Security
# description : Local networks
# navigation  : 1000 1300
```

You might then put something like this in your own script:

```
# heading     : Security
# description : Advanced security
# navigation  : 1000 1250
```

> **Tip:** When naming your script, use a name which closely resembles the description (and hence the name in the navigation panel). This makes it easier to correlate menu items to Perl scripts. Just take the descriptive name and remove capital letters, punctuation and spaces. For instance, "Advanced security" might become `/etc/e-smith/web/functions/advancedsecurity`

When you have your web manager script working, place it in `rpms/SOURCES/root/etc/e-smith/web/functions`

7. Additional scripts, files, etc.

You may also wish to create additional scripts, files or directories which will enhance your packaged software. An example might be custom logging facilities. As with all previous components of your package, simply put these files in the appropriate place under `rpms/SOURCES/yourpackage-x.y.z/root/`.

If your scripts are to be run by the superuser, and are only applicable to e-smith servers, they should go in `/sbin/e-smith`. Otherwise, put them in the appropriate place for any standard RedHat system – probably `/usr/bin`.

8. Building the RPM

This section describes the process for building an RPM, step by step.

1. Create a tarball of the software you've place in the `SOURCES` directory:

   ```
   cd rpms/SOURCES
   tar -cf yourpackage-x.y.z.tar yourpackage-x.y.z/
   gzip yourpackage.x.y.z.tar
   ```

2. Grab a copy of the sample spec file from http://www.e-smith.org/docs/howto/building-contribs-example.spec
3. Copy the `building-contribs-example.spec` to your own `rpm/SPECS/` directory and name it *`yourpackage-x.y.z`*`.spec`
4. Edit the spec file to reflect the meta-information for your own package. The example spec file contains comments (lines starting with a hash symbol (#) are comments which explain what's going on.
5. Check that your RPM will build OK:

```
rpm -bp yourpackage-x.y.z.spec
```

It should say "exit 0" if it was successful

6. Run the **rpm** command again to actually create your RPM:

   ```
   rpm -ba yourpackage-x.y.z.spec
   ```

7. If everything was successful, the last line of output should be `exit 0`.
8. The RPMs should have been generated and put into `rpms/RPMS/i386/` (for RPMs that are compiled to run on one platform only) or `rpms/RPMS/noarch/` (for RPMs that can run equally well on any platform). A source RPM should also exist in `rpms/SRPMS/`.
9. Test your RPM by installing it on an e-smith test box.

More information about building RPMs can be found at http://www.rpm.org/RPM-HOWTO/build.html. This is especially recommended if you wish to use anything more than the extremely simple outline given above. For instance, you may wish to to build RPMs using original source and patches or include more detail and functionality in your spec file.

9. Distributing your RPM

In the near future e-smith will be releasing an area on the website allowing developers to upload their own RPMs, but in the meantime follow the procedure below.

Post an announcement of your RPM to the devinfo mailing list, and provide a download location for those who would like to try it out. If you would like to make it available from the e-smith website, say so, and we’ll put it up there.