The e-smith server and gateway

Technology and community

Kirrily Robert
e-smith, inc

What is the e-smith server and gateway?

The e-smith server and gateway system is a Linux distribution aimed at small to medium enterprises. It is intended to simplify the process of setting up Internet and file sharing services, and can be administered by a non-technical user with no prior experience of Linux.

Functionality

The main functionality provided by the e-smith server and gateway includes:

- Connection to the Internet via modem or various other methods (DSL, T1, whatever)
- Internet gateway functionality (via NAT)
- Mail server
- WWW server
- HTTP proxy
- FTP server
- File sharing with Windows and Macs
- Web-based email
- Highly extensible architecture

The system is configured via a friendly web interface, called the "e-smith manager".

Shared file space appears to the user in the form of "i-bays". The "i-bay" is a unique feature of the e-smith server and gateway system. It allows users to set up an area to store files in, and to choose how those files should be made available (web, ftp, samba, etc) and to whom they should be made available: the whole world (with or without password protection), or only inside the organisation (with or without password protection). At no point does the e-smith administrator need to work directly with the underlying Unix configuration files.

Core technologies

The e-smith server and gateway is based on a number of widely available Open Source software products:

Table 1. Software used in the e-smith server and gateway

Software	License	Role in e-smith
Red Hat Linux	GPL	base operating system
RPM	GPL	package management for upgrades, add-on modules, etc
Perl 5	GPL/Artistic	configuration file templating, system administration tasks, web-based administration system
Apache	Apache (BSD-like)	web server
qmail	"free beer"	mail transfer agent
fetchmail	GPL	mail transfer agent
obtuse-smtpd	BSD-style	mail transfer agent
Squid	GPL	HTTP proxy
ProFTPd	GPL	FTP server
Samba	GPL	Windows file sharing
netatalk		Macintosh file sharing
Horde/IMP	GPL	Web-based email access

Architecture and design

User choice

Many other Internet server and gateway systems currently available require the use of specialised hardware, either in the form of a vendor-supplied special-purpose box (as in the Cobalt Qube) or by limiting the range of supported hardware.

Similarly, some systems provide for only a limited range of Internet connection types, network services, etc.

One of the aims of the e-smith server and gateway is to handle as wide a range of configurations as possible:

- Almost any commodity Pentium-grade PC
- Wide range of network cards
- Dialup, DSL, cable, T1, PPPoE...
- Permanent connection or not
- Permanent or dynamic DNS
- A range of Dynamic DNS providers supported
- Any kind of service or application (provided someone makes an e-smith contrib module for it)

While some of these options are not formally supported, all of them can be made to work on the e-smith server and gateway platform, and users are actively encouraged to tailor the system to suit their own needs, and to contribute any improvements back to the project.

Policies, not individual config files

One of the ways in which the e-smith server and gateway differs from webmin or many other web-based system administration tools is in the way it allows administrators to set general policies for the server rather than individual configurations for each application.

For instance, an i-bay which is configured for use on the organisation's intranet will automatically cause appropriate configurations (i.e. no access from outside the local network) to be generated for Apache, ProFTPd, Samba, etc.

Coherent modularity

The overall design of the e-smith server and gateway can be described in two words as "coherent modularity".

The system is designed to allow the installation of add-on/contrib packages via RPM. However, these packages all appear in a consistent manner in the e-smith manager, by providing a "panel" (effectively a web page (or set of web pages) embedded within the framed interface) as part of the package. These panels maintain a consistent style, and appear to the user as a coherent whole.

As simple as possible, but no simpler

The design of the e-smith server and gateway shows a definite tendency towards simplicity. Wherever possible, functionality is implemented using simple techniques easily understood by programmers of less than wizardly skill.

For instance, the core configuration is stored in a single text file in the following format:

```
AccessType=dedicated
ExternalDHCP=off
ExternalNetmask=255.255.255.0
```

This is parsed by simple Perl code that anyone could understand, rather than using a more complex (and more failure-prone) database, registry, or other form of storage.

The rest of the system is similarly simple to understand; in fact, the key concepts are documented on the developer website (http://www.e-smith.org/) in a document only 8 (printed) pages long.

While it is not the purpose of this paper to reiterate that document, the following brief list of concepts demonstrates the application of the "KISS" (Keep It Simple, Stupid) principle within e-smith.

- *Master configuration file:* as described above, the master configuration file contains information related to system-wide settings and policy decisions.
- *Account file:* Information about accounts on the system is kept in a similar file, `/home/e-smith/accounts`. This contains more information than `/etc/passwd` would normally contain, including such things as email forwarding, groups belonged to, etc.
- *Configuration templates:* Templates for all the configuration files normally found in `/etc` are kept in `/etc/e-smith/templates`. These are parsed using Perl's `Text::Template` module, which parses template files like the following example, interpreting anything inside braces and putting the returned value in that place.

  ```
  search { $DomainName }
  nameserver 127.0.0.1
  ```

  `Text::Template` can also loop through arrays of data, or interpret other Perl code within the template.

- *Events:* An event is typically triggered by the submission of a form in the e-smith manager. Events include:
  - console-save
  - ibay-create
  - ibay-delete

- ip-change
- manager-misc
- post-install
- user-create
- user-delete

Each of these events is represented by a subdirectory under `/etc/e-smith/events`, which contains symlinks to each of the actions to be triggered by that event.

- *Actions:* Actions are triggered by events. Each action is a Perl script stored in `/etc/e-smith/events/actions`. Examples of actions include:
  - conf-dialup
  - conf-servers
  - create-ibay
  - create-user
  - rebuild-ldap
  - restart-servers
  - update-dns

Building modules for the e-smith server and gateway

Anyone with a reasonable understanding of Linux can start building modules for the e-smith server and gateway. The only additional skills required are the ability to RTFM, and domain-specific knowledge for the application being built. Some knowledge of Red Hat based distributions, RPM, and Perl are also advantageous.

A HOWTO document is available on the e-smith developers website, outlining the steps for building modules. A skeleton RPM is also available.

The following is a very brief outline of the steps involved in building an add-on module for the e-smith server and gateway; for more detailed information, refer to the HOWTO document (the URL for which is listed in the references section at the end of this document).

- Set up your personal RPM environment, to avoid having to be root to build things

- Install the skeletal e-smith package
- Make yourself a new RPM spec file by coping the example in the skeleton
- Develop your code under the `BUILD/` directory
- Create templates (using Text::Template) required by your module
- Create actions (Perl scripts) required by your module
- Create events (subdirectories with symlinks to relevant actions)
- Create "panels" for the web manager
- Make a tarball of your package
- Build binary and source RPMs
- Release!

Building the e-smith community

Part of my work at e-smith is to help build the e-smith community. This mostly means the Open Source developer community, though there are other efforts underway to build communities for end-users, resellers, etc. Interestingly, there is a fair degree of crossover between the end-user, reseller, and developer communities: developers become resellers, resellers start taking part in the development community to better serve their customers, and so on.

Why do we want a community?

- Development contributions (e-smith add-on modules, etc)
- Testing and bug-swatting
- Word of mouth publicity
- Nurture potential resellers
- Discover potential staff

The developer website

The main tool we use to build the developer community is our dot-org website (http://www.e-smith.org/). It provides the following features to developers:

- News and announcements
- Discussion fora
- Bug tracking
- Contrib upload
- Documentation

Mailing lists and web fora, and their patterns of use

We run a couple of mailing lists (in particular, the "devinfo" mailing list for developers) and some web-based fora. Interestingly, it has been found that the web-based fora receive much more traffic than the mailing lists.

It has been suggested that the people using the developer website and mailing lists are not as experienced as the general population of Open Source developers, and that they are not familiar with the common use of mailing lists as a communication tool for development teams.

This suggests to me that the e-smith developer community need nurturing not only as "assets" which e-smith hope to attract and retain, but as developers in their own right. Helping them become better developers and more in tune with the Open Source development community will benefit e-smith immediately, by improving the understanding between e-smith developers and technical staff (many of whom are long-time Unix/Open Source/Internet types) and the quality of contributed software.

The CVS challenge

When I started working with e-smith in October 2000, I was surprised to find that they did not use CVS. To me, and to many other Open Source developers, this is one of the core tools which we expect to find in place in any major project.

As it happens, there are good reasons for e-smith's choice not to use CVS so far: CVS and e-smith's use of RPM as a version tracking tool don't mix well. The typical mode of development for e-smith is to take a standard RPM for a widely-used package (for example, Apache) or an existing e-smith package (such as parts of the e-smith manager interface) then make some modifications. The diffs are included in the SRPM as a patch, thus maintaining a history which shows the original software and the changes e-smith developers have made over time.

In October/November, Dan York and myself started using CVS for the e-smith documentation, which we were converting to Docbook. This was the first step in the direction of CVS.

In December 2000, I discussed CVS with some of the technical team. Benefits of CVS, as I see it, include:

- shallower learning curve for experienced Open Source developers
- the ability to make small changes without having to roll a new RPM
- the ability to easily work remotely

The first benefit is particularly attractive to e-smith as we try to encourage more contributions from the Open Source community.

At the moment, I am researching ways of integrating CVS and the RPM-history-keeping styles of development. One tool which looks promising is "gbuild", which can build tarballs, RPMs, and SRPMs from a CVS tree.

Other community involvement

There are a multitude of other ways in which e-smith is attempting to involve itself in the Open Source community. These include:

- LUG visits
- Announcements on freshmeat etc
- Free CDs

Future plans

The e-smith server and gateway has now reached stability as a software project. We have the necessary infrastructure in place to start working more on add-ons, customisations, and services.

The development of the e-smith server and gateway is now tending away from systems development and towards applications development, with a related shift in the type of skills, experience and interests required of developers. This also permits developers to work on e-smith without a deep understanding of the underlying operating system.

We are hopeful that the coming months will see an increase in contributions from the Linux and Open Source communities, and a range of add-on modules which exceeds the current developers' imaginations.

References

- http://www.e-smith.org/