SipX Setup Wizard Design

Revised: June 14, 2006

Introduction

The “modularity project” for sipX 4.0 has a number of overlapping goals related to simplifying the installation and maintenance of sipX-based products. As part of that work, we are implementing a two-phase install:

- Installation: Get the software onto the machine and do non-interactive installation steps, via RPM install.
- Setup: Do interactive installation steps via the “setup wizard”.

This memo outlines the design of the setup wizard.

Design Constraints

Setup wizard should:

- Ask the minimum number of questions needed to get the system up and running. Config Server is a much richer product that can take over for subsequent configuration.
- Run in a text-only mode on machines that don't have GUI support.
- Use the same UI library as the “anaconda” installer, anticipating that in the future we will offer a CD image install via anaconda integration. That UI library is PyGTK, a Python wrapper around the GTK toolkit.
- Anaconda can run in text-only mode; the setup wizard takes the same approach.

Installation

Initial non-interactive installation puts just two components on the machine:

- sipX **process manager**, which acts as an agent on the machine for sipXconfig, among other duties;
- **yum**, for local or remote installation of components.

Start the process manager and configure it to run whenever the machine boots.

Because yum is specific to Fedora Core and is not a Linux standard, modularize the design to allow other packaging mechanisms like aptget (Debian) to be used instead.

Setup

Define a “cluster” as a set of servers (computers) that comprise a single sipX ECS, managed by a sipXconfig instance running on one of the servers. Call the sipXconfig server “primary” and all other servers “secondary”. Then there are two use cases for the setup wizard: setting up the primary server vs. setting up the secondary servers. To distinguish these two cases, the setup wizard asks the user:

- **Will the ECS will run on more than one server?** [no]
  - If the answer is no, then this is the primary server setup case with a single-box ECS. If the answer is yes, then this is a multi-box ECS. In the multi-box case, the wizard then asks:

- **Will this server run the configuration service and manage the other servers?** [yes]
  - If the answer is yes, then this is the primary server setup case, otherwise it's the secondary server setup case.

Primary Server Setup

Steps:

- **Copy RPMs and set up yum repository**: The installation media contains a full set of sipX RPMs in a compressed package. Open up the package and copy all the RPMs into, say, /var/sipxdata/yum. Set up that directory as a yum repository so that other servers can use this server as a yum source. (Future: optionally retrieve RPMs from Pingtel as a yum source so that it is not necessary to use physical media for the initial installation.)
- **Ask the user for the SIP domain name**, which defaults to `hostname --domain`. Example: if the hostname is `gumby.example.com`, then the SIP domain name defaults to `example.com`.
- **Set the realm** to be the same as the SIP domain name. The realm concept is confusing, don't even tell the user about it. In the unusual case where someone wants to change the realm, they can do it manually, as long as they change the realm before creating the first user (superadmin) in sipXconfig.
- **Install sipXconfig** via a local yum install.
- **Create and install certificates**. Create and install a private CA certificate and a server certificate signed by that CA. Create a certificate signing request (CSR) file as well, in case the user wants later to obtain and install a server certificate signed by a well-known CA like VeriSign. These steps require collecting from the user all the info that the `gen-ssl-keys.sh` script asks for: country name, state or province name, etc. (We don't need to ask for the SIP domain name since that's covered already.) We're providing the same functionality as the scripts `gen-ssl-keys.sh` and `install-cert.sh` , wrapped up in a GUI. (Do we need to restart process manager in order for it to use the certificates?)
- **Launch sipXconfig to finish setup**. Note that none of the sipXconfig UI described here exists today, this is all new work.
- **Through the sipXconfig UI, user chooses the products/components to install**, e.g., full PBX, just sipXconfig, etc.
- **SipXconfig installs the selected products** by doing a local yum install. (Or perhaps for symmetry with what happens on secondary servers, instead it starts the process manager then does the local yum install via a SOAP call to the process manager.)

Secondary Server Setup

Steps (continued from initial installation):

- Setup wizard asks the user for the **fully qualified hostname (FQHN) of the primary server**.
- Assume that the **sipXconfig port number** is the default value, 8443. User can override that by appending "*:port number*" to the FQHN. The vast majority of users won't do this, so don't advertise this detail.
- Tell the user to go **run the "add server" command** on the sipXconfig web UI. (Ideally the user doesn't have to go to a separate window at all, perhaps we could embed the sipXconfig web UI right in the setup wizard frame so that it's all seamless, no window switching?)
- SipXconfig asks the user for the **FQHN of the new server**, then tries to contact the process manager on the new server. Bail out with an informative error message if that fails.
- SipXconfig **generates a certificate** for the new server, then generates a random

password for transferring the certificate.

- SipXconfig presents the password to the user and tells them to go back to the setup wizard and continue.
- Setup wizard asks the user for the password, then fetches the certificate from sipXconfig, using the password as a shared secret for authentication, and **installs the certificate**.
- SipXconfig **updates topology.xml** to add the new server, once the certificate has been retrieved.
- Setup wizard sends the user back to sipXconfig. SipXconfig prompts the user for which products/components to install, then **yum-installs the software** via a remote SOAP command to process manager on the secondary server, using the primary server as a yum repository.