Upgrading BIOS and Firmware

on Linux-based Systems with Firmware-tools

Firmware-tools, an open source development project, aims to simplify the process of upgrading BIOS and firmware on Linux® OS–based systems. This article discusses the firmware-tools architecture, which delivers and installs these upgrades using the native Red Hat® Package Manager (RPM™) format and native Linux change-management frameworks such as Novell® ZENworks® Linux Management and the yum tool.

BY MATT DOMSCH AND MICHAEL E. BROWN

Related Categories:

Linux

Visit www.dell.com/powersolutions for the complete category index.

BIOS and firmware updates can be implemented in various ways depending on the system software environment. These updates can be delivered on floppy disks that boot into a DOS-like environment; however, floppies are slow, limited in capacity, and usually require manual operation, making them difficult to deploy remotely or automatically. System vendors have also produced BIOS and firmware installers that work from within an OS.

Firmware-tools, a Dell-developed open source project, is designed to align the delivery and installation of BIOS and firmware updates with existing Linux packaging and distribution formats. This method may not yet be officially supported by all system vendors, but community participation in project development is welcome.

Understanding the firmware-tools architecture

The firmware-tools project builds on the capabilities of the existing Dell™ Update Packages for Linux—self-contained BIOS and firmware installers for Dell systems running Novell SUSE® Linux Enterprise Server or other Linux operating systems. Each package provides an inventory tool to discover current system information and compare it with the new payload version, an executable tool to install the payload on a flash memory chip, and the payload data. Dell also provides tools to integrate these packages into the Altiris®, Microsoft® Systems Management Server, and Novell ZENworks Linux Management Dell Edition change-management frameworks to help simplify deployment using these platforms.

The primary goal of the firmware-tools project is to integrate BIOS and firmware updates into the same change-management frameworks administrators already use to manage their operating systems, which can shorten the administrative learning curve and help improve efficiency. A secondary goal is to support additional operating systems (for example, OpenSUSE and Fedora) and system types (servers, workstations, desktops, and notebooks) as development resources permit. Finally, splitting the inventory, executable, and payload components into separate packages can help reduce code duplication and thereby also help reduce the amount of data to store and download.

Integrating BIOS and firmware updates into change-management frameworks

Existing Linux change-management frameworks can already handle Red Hat Package Manager (RPM) packages. By packaging BIOS and firmware updates and their related tools into RPM packages, the firmware-tools update functionality can be easily integrated into existing change-management frameworks.

Supporting additional operating systems and system types

The firmware-tools framework is written in Python, which is already included in Linux distributions. Using the firmware-tools framework for common tasks can help reduce the size of per-device-type inventory and executable applications. Dell anticipates that the inventory and executable applications will be compiled and built for each target Linux distribution version, and encourages these applications to be fully open source whenever possible to facilitate use in a wide variety of environments.

The firmware-tools framework also supports multiple target system types: servers, workstations, desktops, and notebooks. In many cases, administrators can use the same software to install BIOS payloads on any of these system types.

Splitting components into separate packages

The firmware-tools architecture splits the inventory, executable, and payload components into separate packages:

- **Inventory packages:** These include applications that report information about the current local system, including system vendor, system type, list of PCI devices, and so on. These packages use a plug-in scheme, allowing them to discover devices that do not fall under one of the standard categories—for example, an application that finds baseboard management controllers specified in System Management BIOS (SMBIOS) or Advanced Configuration and Power Interface (ACPI) tables rather than PCI.
- **Executable packages:** These include applications that install a given payload file into a specific device flash memory part. These packages also use a plug-in scheme, allowing them to easily integrate device-specific executables.
- **Payload packages:** These contain only the payload data files themselves (typically binary files) for a particular device.

The firmware-tools framework, like Dell Update Packages for Linux, can deliver package dependencies within the packages themselves and only requires administrators to download a single package per system or device type: firmware-tools uses the RPM provides/requires language to explicitly state inter-package dependencies, and uses an online distribution mechanism to resolve the dependencies automatically with a single download and install command. These capabilities allow multiple device update packages to use and share components such as the system inventory application, and help reduce code duplication.

If a new inventory application is released to add a feature or fix a problem, only that inventory application needs to be changed and pushed out to target systems, rather than the full self-contained package for each affected platform type. The framework also provides unifying applications such as inventory_firmware and apply_updates to implement runtime ordering of inventory, execution, and conflict resolution and notification for the plug-ins.

Using firmware-tools with change-management frameworks

This section discusses example firmware-tools usage scenarios for two change-management frameworks: Novell ZENworks Linux Management and yum.

Novell ZENworks Linux Management

Novell ZENworks Linux Management provides an easy way to keep Linux systems up-to-date. Novell releases packages containing new features and bug fixes in bundles and catalogs through a ZENworks Linux Management server; target systems check in with this server every few hours to download and install any updates. Firmware-tools places RPM packages containing firmware inventory, executable applications, and payload data into custom bundles on local ZENworks Linux Management servers, which target systems can then subscribe to.

To use firmware-tools with ZENworks Linux Management, administrators must do the following:

1. Obtain the firmware-tools and payload RPM packages and place them in custom bundles and catalogs on the ZENworks Linux Management server.
2. Subscribe the target systems to the custom catalogs.
3. Install the firmware-tools framework, vendor-specific tools, and payloads using the `rug` command.

The "Using firmware-tools with Novell ZENworks Linux Management" sidebar in this article provides example commands to carry out these steps; for additional details, visit the firmware-tools Web site at linux.dell.com/firmware-tools.

Yum

OpenSUSE, the Linux development project sponsored by Novell, uses a software update distribution tool similar to Novell ZENworks Linux Management. Yum (Yellowdog Updater, Modified) allows individuals to publish RPM packages on a Web server. Firmware-tools enables administrators to generate yum repositories containing the payload RPM packages, and systems can then subscribe to these repositories to obtain updated package versions. If yum is configured

USING FIRMWARE-TOOLS WITH NOVELL ZENWORKS LINUX MANAGEMENT

To populate a Novell ZENworks Linux Management server with firmware-tools packages and payloads and deploy those updates to target systems, administrators should first perform the following steps (as the root user) on the ZENworks Linux Management server:

1. Install ZENworks Linux Management on the server.
2. Create bundles using the following commands:

```
# zlman bundle-create bundle-dell-software
# zlman bundle-create bundle-dell-firmware
```

3. Obtain the RPM packages from the firmware-tools Web site.
4. Add these packages to the bundles, once per desired os-version-arch target:

```
# zlman bundle-add-package --installtype=upgrade
    bundle-dell-software sles-10-x86_64
    packagename.rpm
# zlman bundle-add-package --installtype=install
    bundle-dell-firmware sles-10-x86_64
    packagename.rpm
```

5. Create a catalog:

```
# zlman catalog-create catalog-firmware-tools
```

6. Add both bundles to the catalog:

```
# zlman catalog-add-bundle catalog-firmware-
    tools bundle-dell-software bundle-dell-
    firmware
```

Next, administrators (as the root user) should install the ZENworks Linux Management client on the target systems using the following commands:

```
# rug service-add https://zlmserver/
# rug subscribe catalog-firmware-tools
# rug install firmware-addon-dell
# rug install $(rug --terse what-provides
    "$(inventory_firmware -b)" | \
  awk -F \| '{print $3}')
# apply_updates
```

Note: Some versions of the `rug` command include a `solvedeps` function, which may be used instead of the second `rug install` line:

```
# rug solvedeps "$(inventory_firmware -b)"
```

to run automatically every night, it can download new inventory, executable, and payload components within hours of their release; administrators can also download the updates manually.

To use firmware-tools with a yum repository, administrators must do the following:

1. Obtain the firmware-tools and payload RPM packages and place them in a yum repository.
2. Subscribe the target systems to the yum repository.
3. Install the firmware-tools framework, vendor-specific tools, and payloads using the `yum install` command.

For more information about these steps, visit the firmware-tools Web site at linux.dell.com/firmware-tools.

Advancing the development of firmware-tools

Firmware-tools is designed to help significantly reduce the administrative burden of updating system BIOS and firmware versions on Linux-based systems. Its architecture has been demonstrated to work with Dell system BIOSs and Dell PowerEdge™ Expandable RAID Controller (PERC) firmware. A goal for the future is to expand the scope of products using this framework to include system types from other vendors, additional device types, and additional Linux and UNIX® OS types, along with additional packaging formats such as those used by Debian, Gentoo, and FreeBSD. Developers and system administrators alike are invited to join in this open source effort.

Matt Domsch is a Linux software architect at Dell and a community member of the Fedora Project Board. Matt has a B.S. in Computer Science and Engineering from the Massachusetts Institute of Technology and an M.S. in Computer Science from Vanderbilt University.

Michael E. Brown is a software developer and open source advocate on the Linux Engineering team at Dell. He was previously the lead developer for the Dell OpenManage™ Server Assistant application, where he led the switch to use Linux as the base OS for that product.

FOR MORE INFORMATION

Firmware-tools:
linux.dell.com/firmware-tools