montavista™

MontaVista Linux 6

TECHNICAL BRIEF

The MontaVista Linux 6 Technical Brief provides a technical overview of MontaVista Linux 6 (MVL6) and each of its components including Market Specific Distributions (MSD), the Software Development Kit (SDK), and the MontaVista Zone Content Server. The Technical Brief concludes with an overview of MontaVista support and maintenance, and a glossary of new terms and definitions important to understand when learning about MVL6.

Table of Contents

MontaVista Linux 6: The New Approach to Embedded Linux Development

MontaVista Linux 6 meets embedded developers where they are in the development cycle with a complete embedded Linux distribution and developer tools for a faster time to development. With Market Specific Distributions, the MontaVista Integration Platform, and unprecedented flexibility in a commercial solution, MontaVista Linux 6 enables developers to build from source to more easily customize their software stack and add product-differentiating features.

The MontaVista Linux 6 Technical Brief provides a technical overview of MontaVista Linux 6 (MVL6) and each of its components, including Market Specific Distributions (MSD), the Software Development Kit (SDK), and the MontaVista Zone Content Server. The Technical Brief concludes with an overview of MontaVista support and maintenance and a glossary of new terms and definitions important to understand when learning about MVL6.

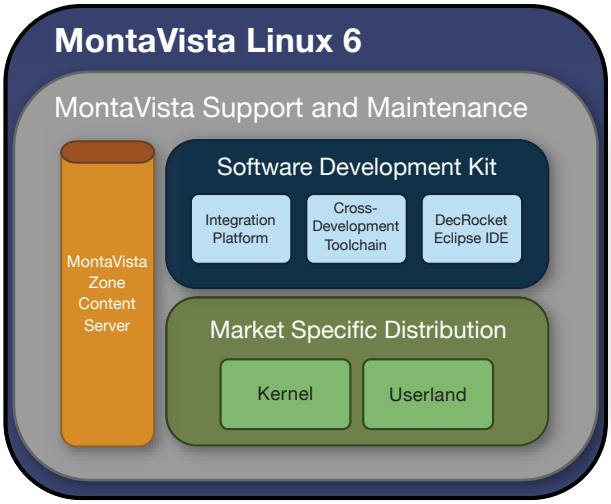

Figure 1 - MontaVista Linux 6

Market Specific Distributions

MontaVista Linux 6 introduces a powerful new approach to embedded Linux system design that benefits developers like never before, Market Specific Distributions. Traditionally companies like MontaVista have modeled their products similar to the RTOS world. RTOS vendors generate “board support packages” (BSPs) that adapt a predefined product as little as possible to execute on a new hardware platform. This model was sensible in the era of proprietary software platforms because the vendor was the only true innovator in the software supply chain.

The open source revolution has changed the embedded software supply chain. Now a worldwide ecosystem of developers contributes to Linux. Semiconductor and processor vendors such as Intel, Texas Instruments, Freescale, ARM, MIPS, and others have shifted strategies and now actively innovate within the open source process to enhance Linux. They do this to showcase their newest products running reference Linux implementations.

Our new MSD approach simplifies the process of selecting and transitioning to a Linux commercialization partner such as MontaVista.

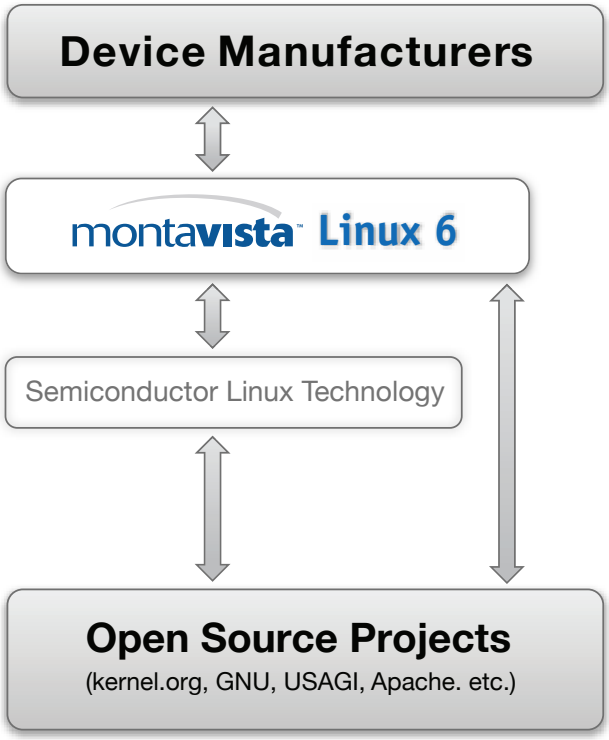

Figure 2 - MVL6 aligns the embedded Linux supply chain to simplify the transition to a Linux commercialization partner

What is a Market Specific Distribution?

Each MSD is a specialized set of Linux technology that includes a Linux kernel, software libraries, and other applications that are collectively referred to as a “distribution.” In the enterprise software world Linux distributions are typically intended for general purpose usage. This isn’t true however in the embedded market. Embedded software developers face unique market demands and design challenges that drive the adoption of specialized hardware platforms.

Each MontaVista Linux 6 MSD is designed to deliver the appropriate Linux technology for the target hardware platform and the hardware’s intended market application(s).

Feature Compatible with Semiconductor Linux Technology

Most development projects begin as an in-house prototyping effort to prove the technology assumptions and justify continued funding. During this period, developers often begin their work using Linux technology from their semiconductor partners, and often within an emulation environment on prototype silicon reference platforms. These reference

implementations are typically well integrated with the hardware, but lack many non-hardware specific features that may be required to deliver a complete commercial product. Technical support and updates on these reference implementations are often extended only to the top customers of each new semiconductor product.

When it comes time to transition to an embedded Linux commercialization partner MontaVista can help. The MontaVista Linux 6 Market Specific Distributions are engineered to be feature compatible with the Linux technology produced by our semiconductor partners. Developers selecting MontaVista should be able to easily transition their applications and device drivers to MontaVista Linux. Each MSD uses the same or a newer base kernel version in order to assure feature compatibility.

Aligned with Your Design Needs

Modern System-on-chip (SOC) designs often involve a complex software stack that extends far above the Linux kernel. The new MSD approach, pioneered by MontaVista, ensures developers have the right software (kernel, drivers, and userland) necessary to get the most out of their selected hardware. In some cases this may mean testing with proprietary audio/video codecs or delivering unique virtualization technologies that exploit the hardware architecture. The MontaVista MSD design process delivers, or is compatible with, key supporting technologies that are typically excluded from a generic Linux but necessary to meet commercial embedded requirements.

Pre-Integrated and Tested

Developer time spent integrating individual components, fixing homegrown tools, or backporting features adversely impacts project schedules, reduces developer productivity, and ultimately adds nothing to differentiate the device. The MontaVista MSD approach selects and integrates a broad array of popular open source software that most projects require. Our quality assurance practices identify defects and fixes them before they can impact project schedules.

MontaVista Software Development Kit

The MVL6 Software Development Kit (SDK) brings together all the essential tools for successful embedded Linux product design that every developer needs to work efficiently. These include:

- The MontaVista Integration Platform (MVIP)
- MontaVista DevRocket Integrated Development Environment (IDE)
- Complete toolchain that includes a set of GNU compilers and utilities, cross compilers for non-Linux runtime environments, and on-target key system libraries

The MVL6 SDK ensures developers have a high quality, pre-integrated set of development tools that can quickly be deployed to engineering workstations and generate repeatable results.

SDK Host Support

Developers often must plan for a decade or more of engineering and maintenance on a new product release. It is critical that the SDK used in their design have broad host support and minimal dependencies upon the host that may impair compatibility with future host operating system versions. Since no company can anticipate what changes may be introduced in future host operating systems, MontaVista has adhered to available standards in order to best position the MVL6 SDK for future host operating system compatibility.

The MontaVista Linux 6 SDK is engineered to only rely upon Linux operating system components and APIs defined by the Linux Standard Base (LSB) 3.1 standard[1]. While not officially supported, current and future host operating systems compliant with the LSB 3.1 standard have an improved likelihood of successful execution of the MontaVista Linux 6 SDK.

MontaVista Linux 6 is tested and officially supported on current versions of Ubuntu, SuSE, and Red Hat Enterprise Linux.

Development Host	Processor
Red Hat Enterprise Linux 5.3	X86, 32 and 64 bit
openSUSE 11.1	X86, 32 and 64 bit
Ubuntu Desktop 9.04	X86, 32 and 64 bit

Table 1 - MontaVista Linux 6 tested host distributions

Host Packages

The MontaVista Linux 6 SDK includes various host packages which are compiled and execute on the development hosts listed above. Each of these packages is intended to help support the execution of the SDK and its primary functions: building software and images via the bitbake command and execution of the GNU Compiler Collection utilities.

Target Packages

Target packages are provided for development via the delivery of multiple collections. Each collection provides numerous recipes that can access original source code from the MontaVista Zone, patches that modify the original source code, and automated instructions to build and create the resulting binaries and optional binary packages. Each recipe may create many binary packages so that developers can easily add or remove components they require in their final system images.

For more information about collections and recipes, see the MontaVista Integration Platform section below.

1 Consult the Linux Standard Base website for more information at http://www.linuxfoundation.org/collaborate/workgroups/lsb

Cross and Native Development Toolchains

Developers preparing their own toolchains often fail to properly isolate their host development system from their cross compilation tools. When this occurs, the symptoms can be subtle and not detected until late in the development cycle. Common issues include cross compilers that cannot be run on alternate development hosts, or target software improperly utilizing host kernel and library headers. Some developers even resort to never upgrading their host development environment for fear that the upgrade will break their commercial products. These typical problems are easily avoided by using a high quality cross development toolchain, like the one found in MVL6, which properly isolates the deployed target software from the development environment.

MontaVista cross development toolchains can generate optimized code for the following processor architectures[2]:

CPU	Configuration
ARMv5T	Little-Endian, ARM
ARMv6	Little-Endian, ARM, VFP
ARMv7-A	Little-Endian, Thumb 2, VFP
MIPS32	Big-Endian, O32
MIPS32	Big-Endian, Soft Float, O32
MIPS32	Little-Endian, O32
MIPS32	Little-Endian, Soft Float, O32
MIPS64	Big-Endian, Soft Float, N32
MIPS64	Big-Endian, Soft Float, N64
MIPS64	Big-Endian, Hard Float, N32
MIPS64	Big-Endian, Hard Float, N64
MIPS64	Big-Endian, simple exec
Power 603	32-bit, Soft-Float
Power 603	32-bit, Hard-Float
Power e500mc	32-bit
Power e500mc	32-bit, bare metal
Power e500v2	32-bit
Power e600	32bit, Altivec
PentiumPro	32-bit
AMD64/Intel 64	64-bit

Table 2 - MontaVista toolchain optimizations

Non-Linux Runtime Environments

Select MSDs include support for non-Linux runtime environments. These non-Linux runtime environments are typically used for dedicated execution of small applications linked to a simple utility library that provides basic services. The SDK for these MSDs includes an additional toolchain that targets this environment and includes the required target C library. The MontaVista Linux 6 SDK ensures that both the Linux and non-Linux runtime environments utilize the same ABI format and a common set of toolchain versions ensuring developer efficiency.

Command Line Debugging Using GDB

Developers often turn to familiar tools for addressing their most common challenges. The GNU Project Debugger, commonly known as GDB, is the command line source level debugging tool most commonly turned to by developers targeting Linux. Table 2 shows the GDB usage for running the debugger in particular modes of MVL6.

Mode	Usage
Cross, Remote gdbserver, Live application debug	Debugging of currently executing userspace applications
Cross, Remote KGDB, Live kernel debug	Debugging of currently executing core kernel or kernel modules
Cross, Core-file, Post-mortem application debug	Debugging of failed applications which have created a core file
Native, ptrace, Live application debug	On-target debugging of running applications
Native, Core-file, Post-mortem application debug	Debugging of failed applications which have created a core file

Table 3 - GNU Project Debugger (GDB) modes and usages in MontaVista Linux 6

On-Target System Libraries and System Size Reduction

Applications running on MontaVista Linux can utilize the core library components provided by the SDK to support C and C++ development. These include the GNU C and C++ libraries (glibc and libstdc++) plus additional ancillary libraries typical for Linux systems. MontaVista utilizes a specialized version of the GNU C and C++ libraries intended for embedded systems. The "Embedded GLIBC" (EGLIBC)[3] library has several key advantages over other alternative size reduced C libraries:

- EGLIBC is available on all architectures supported by GLIBC
- EGLIBC supports both C and C++
- EGLIBC is quality software based on decades of effort invested in GLIBC
- The MontaVista EGLIBC based SDK can be deployed in 54% of the storage space typically required by standard GLIBC[4]

2 Each MSD is shipped with the appropriate toolchain for the intended hardware architecture. Not all MSDs support all available toolchain configurations.

3 For further information on EGLIBC consult the EGLIBC project website at http://www.eglibc.org/home

4 Based on MIPS architecture with all optional components disabled.

MontaVista is leading the market by providing the highest quality C and C++ library code in a size-reduced format that does not impinge upon application compatibility, reliability, or performance. EGLIBC has proven compatibility with GLIBC as evidenced by the Debian project's recent adoption of EGLIBC as the default C/C++ library for the Debian Linux distribution.

MontaVista Integration Platform

The MontaVista Integration Platform (MVIP), and included build system, is the centerpiece of the SDK. The MVIP creates the original MSDs at install and empowers developers to adapt MontaVista Linux to the needs of their design. The MVIP enables source-driven customization of the entire MontaVista Linux-based software stack and can assist developers with the following features:

- Based on de facto standards and an open source core
- Always provided as original source + patches
- Build and rebuild entire product lines with one tool
- Structure customizations with collections
- Build-compatible with thousands of community packages
- Easily extensible and compatible with automated build
- Easy to upgrade and manage change
- Complements existing SCM systems

Let's look at each of these areas in more detail.

Based on De Facto Standards and an Open Source Core

At the heart of every embedded Linux development project, the build system is relied upon daily. Therefore it is imperative the build system be reliable and flexible in order to address unforeseen requirements. Build systems are also inherently difficult to change once integrated into the daily developer workflow and supporting automation. For such a critical need, developers should only rely on tools with a proven heritage of satisfying requirements similar to their own. Using these criteria, MontaVista selected the BitBake[5] utility as the core for the MontaVista Integration Platform. The BitBake utility falls under the umbrella OpenEmbedded[6] (OE) project and has been a key enabling technology of the OE project since 2004.

The BitBake-powered MVIP benefits developers by ensuring they do not have to invest in learning a proprietary, vendor-specific technology that may not be flexible enough to accommodate future requirements.

Always Original Source + Patches

MontaVista Linux 6 is a source-driven product that empowers developers to maintain a stable software base and integrate changes from open source, or local customizations, in order to deliver products to market. The source code that constitutes MVL6 is always distributed to the customer in the form of the original unmodified source code archive from the upstream project plus any patches that might have been applied in order to resolve defects or improve the software.

The MVIP can catalog all of the original source code and patches that contribute to a customer design and prepare them for distribution. This "software bill of materials" can be used to help comply with applicable open source licenses by providing development engineers with a complete record of all source code included in their product.

Build Your Entire Product Line with One Tool

Customers don't build just one application or product. The challenges of commercial product development require managing entire product lines with multiple configurations and maintenance levels. Designing complex Linux-based embedded systems could mean managing 10+ million lines of source code. Developers must be able to ensure all changes to one product configuration are inherited from that version to all of its derivatives.

Simply dropping all of this code into an isolated Software Configuration Management (aka Source Control Management) system like CVS, ClearCase, or Subversion isn't going to suffice because the code will be updated periodically. The open source community is continuously submitting changes, and partners that contribute to your product development evolve their code as well.

Intermingling community, semiconductor partner, commercial Linux vendor, and local development team patches into one development stream inhibits progress and can result in confusion when components are upgraded independently.

By integrating with your SCM system, tracking changes to community code, and managing dependencies, the MVIP helps you manage this constantly changing process, ensuring you have successful, repeatable builds.

Structure Your Customizations with Collections

The MontaVista Integration Platform's powerful task and collection[7] powered system can use shared collections for common elements while extending or replacing software elements that are unique to one offering in the product line. The shared collections and inherited configuration settings save valuable time across projects through reduced update and maintenance overhead while improving build performance.

5 Consult the "BitBake User's Manual" at http://bitbake.berlios.de/manual/ for additional technical details on BitBake

6 See http://www.openembedded.net/ for more information on OpenEmbedded

7 See end Glossary for definitions to commonly used terms in MVL6

Collections ensure locally generated software contributions are isolated from the evolving base of Linux software which together represents your product design. Ultimately, collections save time and reduce repetitive work by helping developers more easily structure and maintain source code and customizations.

MontaVista MSDs are fabricated out of one or more collections. Each collection can introduce new software packages, patches, or configuration changes to the software stack build. Collections are stacked on top of each other and each collection can replace software packages from lower components.

Each MVL6 MSD is typically composed of the Foundation, Core, and one MSD collection. Depending on the hardware capabilities other collections may be included as well. Current collections include[8]:

Name	Summary	Recipes
Foundation	The essential development host and target recipes required for building and running a small Linux system. The majority of these support building the small set of target packages provided	100
Core	An expanded set of recipes for expanded Linux system functionality	160
MSD	Each MSD is delivered with a collection named after the MSD containing the Linux kernel. May also include other MSD specific software not bundled with other collections	1
Audio	Audio recording, playback, and manipulation	3
Bluetooth	Support software for Bluetooth radio support	2
Graphics	Base X.org support and programming libraries	40
Wireless	Wireless networking utilities	4

Table 4 - Example collections associated with MontaVista Linux 6

Build-Compatible with Thousands of Community Packages

As mentioned, developers often must supplement the software packages available from MontaVista with additional packages from the open source community. With the MontaVista Integration Platform this common practice is now simplified. The MVIP is compatible with the recipe format used by the OpenEmbedded community. Developers can now select from over 6000 packages via OpenEmbedded and add them to their MVIP managed projects.[9] Refer to Appendix B for an example of an OpenEmbedded recipe.

8 Specific MSD's may require alternate or upgrade versions of the components listed. Recipe counts are approximate and will change over time.

9 MontaVista standard technical support applies only to packages distributed as part of a MontaVista created collection in a MSD. Professional services are available to support other available software packages.

Easily Extensible and Compatible with Automated Builds

Engineering best-practices recommend continuous and automated build systems be used for ensuring repeatable builds and monitoring software quality. This time saving ability enables teams to leverage off-hours to run systems 24 hours a day. Developers have full control of the MVIP directly from the command line and they can introduce their own extensions by utilizing the Python scripting language. This lets them create new tasks that are automated and integrated into the build environments.

Easy to Upgrade and Manage Change

With developers enhancing and maintaining multiple commercial products, efficient change management is an essential competency. Simplistic techniques, like checking the Linux kernel source code into a revision control system, cannot ensure the entire product line is maintainable, reproducible, and can be enhanced without breaking other builds. With the MVIP developers can:

- Control the introduction of unplanned software updates by "locking down" the build configuration
- Generate test builds of new configurations that can be rolled back to known good configurations
- Work cooperatively with your revision control system to manage the entire software stack configuration, sources, and
- customizations

Complements and Integrates with Your Existing SCM System

Most companies have standardized on a corporate Software Configuration Management (SCM) system for managing the evolution of their software assets. The MVIP can be used with many SCM systems, including popular commercial and open source options (i.e. CVS, Subversion, and GIT), by simply controlling updates to your project directory and archiving the original source code and metadata collections. This clear methodology assures developers their software build components are fully archived and their builds will be reproducible in the future.

For larger teams that may use multiple SCM systems, the MVIP has the flexibility to directly access more than one system in order to pull code directly into the build. Each software package that contributes to the build can be fetched from a defined SCM. For added convenience, source code, binaries, and patches stored in compressed tarballs can also be fetched via HTTP/HTTPS or from the local file system. The MVIP is meant to seamlessly integrate into your existing development environment, not disrupt it.

MontaVista DevRocket 6 IDE

MontaVista DevRocket 6 is the Integrated Development Environment (IDE) that supports MontaVista Linux 6. DevRocket 6 delivers a set of tools designed to streamline and automate common embedded Linux development and analysis[10] tasks. DevRocket 6 is based on the Eclipse project and is delivered as a set of Eclipse plug-ins that increase developer productivity by simplifying the complex tasks of embedded Linux development. DevRocket 6 plug-ins can work within standard Eclipse-based platforms based on the Ganymede release or with the bundled Eclipse runtime delivered with MVL6.

New in MontaVista DevRocket 6: MVIP Projects

The new MontaVista Integration Platform provides developers with a powerful new command-line based tool for customizing and compiling an entire Linux based software stack. DevRocket 6 introduces a new project type that exploits the power of the MVIP from the DevRocket graphical user interface.

New in MontaVista DevRocket 6: MemTraq

Developers building Linux based products often struggle to find memory leaks in deployed systems. Traditional techniques such as Valgrind or mpatrol are limited because they require:

- excessive CPU overhead
- special runtime configurations that can't be used in production deployed systems
- on-target storage for memory trace information

To address these limitations MontaVista created a new memory leak analysis framework for MontaVista Linux 6 that integrates at the lowest level of the userspace software stack. This new feature, known as MemTraq, provides developers with visual depiction of live memory leak information from running applications without disrupting operation of the program under analysis. Unlike alternative solutions, MemTraq does not require invasive binary patching of applications while running and can operate without slowing down the application being analyzed. These enhancements enable developers to gain efficiencies in tracking down memory leaks in order to deliver devices to market with higher quality.

Target Management

DevRocket 6 utilizes the open source Eclipse target management project called Remote Systems Explorer (RSE) that includes a full terminal interface used to log in to and run commands on remote targets. MontaVista created and contributed back to the community an SSH implementation for RSE that allows target management on any MontaVista Linux target using the industry standard SSH protocol to support a wide range of target services that otherwise would need to be manually set up by the developer. These prerequisite services include file and process management, remote terminal/shell, and fully automated debugging and analysis.

Fully Automated Edit/Compile/Debug

MontaVista DevRocket accelerates time to market by delivering a streamlined and fully automated edit/compile/debug cycle, thus eliminating the multiple manual and error-prone steps involved in building binaries, copying them to a target, launching the debug server, and connecting back to the host. DevRocket 6 manages deploying applications to the target and identifying the toolchain to construct the correct debug chain for your project. Multiply these time-saving steps across an entire team and it can really affect a products time-to-market.

Advanced Analysis

DevRocket 6 delivers intuitive, interactive, and accessible interfaces to configure, manage, execute, and present results from best-of-breed FOSS analysis tools. Significant productivity gains are realized when solving common analysis questions such as:

- How much memory is my system using and which components are responsible for it?
- Where are my system and application performance bottlenecks?
- What is the source of my memory leak?
- Which events have transpired on the system and why?

MontaVista Zone

The MontaVista Zone is an exclusive online support site available to all current valid MontaVista subscribers.

The MontaVista Zone provides 24x7 access to:

- Downloads for the latest software releases, updates, and up grades
- Online documentation
- Frequently Asked Questions (FAQs)
- Interactive online tutorials
- Open and closed known problem reports (with the resolution for each closed problem report)
- RSS feed for notification of all patches/updates
- Proactive security monitoring and patch distribution
- Detailed list of all hosts and targets supported, by architecture
- Detailed information that covers:
    1. Development Environment
    2. Kernel Information
    3. Application Development
    4. Integration/Deployment

10 Debug and analysis tools are dependent upon kernel and userspace features implemented in the Market Specific Distribution. Consult the MSD documentation to see if the required features are implemented and supported.

Figure 4 – The MontaVista Zone provides a knowledge base and access to software and updates

Integration with MontaVista Linux 6

With MontaVista Linux 6, the MontaVista Zone is now directly integrated with the MontaVista Integration Platform. This new integration with the MontaVista Zone Content Server enables developers to quickly access software updates and archives without having to manually download individual product components, thus saving time and increasing efficiency. Using a single command, developers update their project or local content mirrors to be current.

Source Mirroring

The open source projects that developers typically use to build embedded Linux based devices distribute their source code via hundreds of individual Web sites. Simply obtaining current copies of all of the relevant source code can take a significant amount of time as each website may be experiencing technical difficulties or have been reorganized. Based on our own experience, it can take days to locate the individual source locations and download all appropriate content, assuming the sites still remain online and contain the appropriate links.

The MontaVista Zone Content Server provides a source code mirror of all required source code components for building MontaVista Linux 6. This source code is automatically accessed by the MontaVista Integration platform as needed during the build process. The source code is referenced in such a way that in the unlikely event the MontaVista Zone is unavailable the code can be automatically retrieved from the original community maintained website. The source mirror function of the MontaVista Zone ensures customers can always access the source code that contributed to their projects. The mirroring and fallback capabilities ensure you will always be able to create repeatable builds.

Prebuilt Staging Packages

Building an entire Linux, based design from source can take hours. Developers shouldn't have to wait while unmodified components are built from source again and again. The MontaVista Integration Platform has extended the BitBake utility to generate and use prebuilt staging packages. These staging packages hold all of the intermediary products and outputs of a build in a form that can be distributed and shared.

Prebuilt staging packages are used to accelerate the initial usage of the product and helps developers quickly get started without waiting for an initial build from source. As developers start to configure their Linux system the prebuilt staging packages are rebuilt incrementally to match the new configuration. These new prebuilt staging packages can be shared amongst the development team (via a network filesystem) to ensure that developers never have to wait on redundant builds.

Local Mirroring and Proxy Support

Many companies have policies restricting direct access to the Internet from software build machines. For those that do the MVIP and MontaVista Zone have two capabilities that can help:

- **Proxy Support:** The MVIP can access the MontaVista Zone via a local web proxy and the HTTP protocol. For open source projects that are available only as Git, CVS, or Subversion repositories the MontaVista Zone contains snapshot tarballs of the repository
- **Local Mirrors:** The MVIP can help you create and update your own local mirrors of the MontaVista Zone Content Server that is used by the MVIP. This will include a snapshot of all content collections, source code, and prebuilt staging packages. The local mirror can then be redistributed via a local network or physical media

Customer Support, Quality Assurance, and Training

Ideal project outcomes depend on more than just source code and compilers. MontaVista has assembled a broad array of support, maintenance, quality assurance practices, professional services, and training offerings so developers can focus on product development.

Support and Maintenance

Throughout the development cycle it's common for teams to burn cycles fixing bugs and resolving issues. This doesn't account for the number of unknown bugs that surface after shipment. As project development ends, customers often transition their efforts to a support and maintenance team. This team is responsible for ensuring critical defects and security risks are addressed on a timely basis.

Support and maintenance developers can use the MontaVista Zone to inspect the most recent defect resolutions provided for their version of MontaVista Linux and integrate these fixes. The MontaVista technical support team is available to help development engineers integrate maintenance changes into their projects.

Technical Support

MontaVista Technical Support provides customers with a complete range of technical engineering support for MontaVista products. Our technical support consultant engineers are highly qualified Linux developers, with hundreds of years of combined experience in RTOS/Linux development, implementation, and support.

MontaVista maintenance and support covers MVL6 components including the kernel, cross compilers, debuggers, host and target applications, and configuration, development and debugging tools. Additional information on specific features supported is available on the MontaVista Zone.

In its core support offerings, MontaVista provides multiple tiers of technical support, designed to meet the individual needs of customers' development teams. Each tier provides both email and telephone based support to one or more named contacts in the license holder organization. Specific tiered offerings include Standard, Premium, Managed, or Dedicated support.

Long Term Maintenance

Customers who require support beyond the end of life of core support can sign up for Long Term Maintenance. Long Term Maintenance is designed for the needs of development teams in industries such as Telecommunications, Transportation, Aerospace, and Defense, where it is common to find embedded equipment with fielded lifetimes of 10 years or more. Customers covered by Long Term Maintenance receive critical and security bug fixes for a fixed version of the MontaVista Linux kernel, OS platform, and development tools, enabling them to focus on other important aspects of their projects. Long Term Maintenance packages are renewable annually or on a fixed term basis.

Product Updates and Security Defect Resolution

The MontaVista Zone provides customers with a stream of product updates that resolve identified software defects and security issues. The product updates are generated based on customer feedback and shared broadly with all customers that have purchased an applicable product. Security defect resolutions typically result from the proactive monitoring of security defects identified in the open source community and by organizations that register defects with the Common Vulnerabilities and Exposures (CVE) database.

Customers can monitor available defect resolutions via MontaVista Zone, RSS, and direct email.

Product Update Model

MontaVista Linux 6 introduces a new product update model that streamlines the update of designs and ensures customers have the most up-to-date software available for their builds. Product updates are distributed as collection updates directly to the MontaVista Integration Platform or the customer's local content mirror. Each update is cumulative and includes all previous versions of each updated software package. These cumulative updates can be safely distributed to developers without disturbing locked-down product build configurations. When developers wish to experiment with deploying updated software components the version lock can be released and the new build will contain the fully updated software stack. Developers can also individually control product updates so that only updates deemed critical are applied to their designs.

Quality Assurance Practices

Embedded Linux is MontaVista's core expertise. Our quality assurance program is built upon a decade of experience testing and improving upon embedded Linux. The foundation of MontaVista quality assurance efforts begins with our people and the procedures we have created. While test automation is used extensively at MontaVista, experience has proven that automated testing alone cannot identify all failure modes.

MontaVista quality assurance practices are performed in stages:

- **Functional testing:** Performed by the expert engineer respon sible for the creation of a new feature or hardware port of MontaVista Linux. This testing effort is focused on assuring the correct functional implementation of the feature, establishing performance expectations, and documenting typical and edge-case operational modes for further testing
- **Automated build testing:** Once the enhancement has been propagated to the common source code repositories the new feature is integrated with the automated build process. Automat ed building typically exposes undocumented dependencies upon the primary engineer's development environment. Removing these undocumented dependencies is critical for assuring that customers can reliably build their MontaVista Linux based prod ucts on any supported development host
- **Automated testing:** MontaVista uses a custom developed, automated test facility to install, configure, test, and analyze the automated builds. This multi-site test facility utilizes over 350 host systems, target boards, and test equipment components. Customized databases and reporting engines track test prog ress and collect results for inspection. Over the years MontaVista has created a suite of test scenarios that utilize available com munity created tests and custom written test scripts

- **Manual testing:** Manual testing is still required for certain types of features and general product quality issues. Developers test I/O device compatibility, general performance, and operation during applicable actions such as insertion and removal. Develop ers also perform installation testing and walk-through testing to ensure that the product documentation matches the actual product usage

By the time testing is completed, over 25,000 individual tests will have been run on the kernel and toolchain. Currently, MontaVista has between 30 and 50 automated test suites and an additional 10 manual test suites that go into improving MVL6 quality.

MontaVista's custom test procedures have been designed to focus attention on the areas most likely to identify latent defects not caught during functional testing. Test procedures are customized for each MSD.

For a list of the types of test procedures run see Appendix A.

Training

MontaVista customer education provides world-class education and training in developing intelligent device applications using MontaVista Linux. With the skills acquired, development teams will be able to reduce application development cycles while minimizing development risk.

Available training topics include:

- Embedded Linux Foundation for Managers
- System Development Jumpstart using MontaVista Linux
- Device Drivers with MontaVista Linux
- Custom On-Site Courses - MontaVista provides courses tuned to customers' requirements and present them at their facilities

Professional Services

MontaVista Professional Services is highly experienced with operating system internals, have in-depth knowledge of the Linux kernel, and can provide integration services across the full software stack, including middleware and applications. MontaVista professional services provide complete, end-to-end assistance to help Linux development teams get their deployments to market rapidly and with less development risk.

The Meld Embedded Linux Community

Meld is a free community designed for embedded developers using Linux to build commercial products. Sponsored by MontaVista, community members contribute their time and advice to help support their peers. You can find Meld at http://meld.mvista.com.

Summary

MontaVista Linux 6 delivers a new approach to embedded Linux design. By providing source based, Market Specific Distributions, along with the MontaVista Integration platform and the other SDK components, developers have new flexibility in their approach to embedded design. For the first time developers can fully leverage the open source community and the semiconductor Linux technology, while still gaining all the benefits of commercial embedded Linux, to deliver better products to market faster, and at a lower cost.

Glossary of Terms and Definitions

BitBake - The MontaVista Integration Platform is built upon BitBake, a global build manager with the ability to resolve dependencies, fetch content from the MontaVista Zone Content Server (or a mirror), and build that content into bootable images.

Classes can be used to define actions that are common for a large number of recipes. For instance, a class can be used to define the common build actions for software based on the open source autotools build environment. Individual package recipes can simply include a class by reference, and replace or extend the default actions only where necessary.

Collections - Groupings of related components. In general, the kernel and its components are grouped as one collection. User space and custom components are grouped into several other collections and included or removed as needed.

Content Mirrors - A mirror is an organization's local copy of one or more MSDs exactly as they appear on the MontaVista Zone Content Server. Development projects can be created using the mirror instead of the Content Server, saving network bandwidth and ensuring that all developers are working from a common base.

Dependencies can be used to declare that a given package requires the support of another package, either at build time or at run time. The build process can ensure build time dependencies are built before the dependent packages, and that run time dependencies are automatically included in any resulting images.

Images are recipes that define deployable outputs from the build. These outputs can include bootloader and kernel binaries, as well as filesystem images containing system and user application software.

Market Specific Distribution (MSD) - A set of collections represents a Market-Specific Distribution (MSD), literally a distribution geared toward a specific market. Each MSD is a specialized set of Linux technology including a Linux kernel, software libraries, and other applications that are collectively referred to as a "distribution." Each MSD is customized to deliver the appropriate Linux technology for the target hardware platform and its intended market application.

MontaVista Software Development Kit (SDK) – The SDK consists of following tools necessary for successful embedded Linux based product design developers need to work efficiently: The MontaVista Integration Platform, DevRocket integrated development environment, and cross development toolchains.

MontaVista Integration Platform (MVIP) - Tools that access the MontaVista Zone Content Server and authenticate themselves to the server. Authentication enables the server to display the content specifically available to the caller. The tools that drive this process and manage the content are collectively called the MontaVista Integration Platform.

Recipes are BitBake script files that define how to build a particular target object. Actions can be defined for various stages of the build process, including:

- fetch: downloads the base source archives
- patch: applies patches containing modifications to the base source archives
- configure: configures build settings for a particular package
- compile: compiles the software.
- install: creates the desired installation structure for the built package contents
- package: bundles the installed package contents into archives
- clean: removes temporary files created by the build process

Tasks are recipes that are associated with no unique software; they contain only dependencies and sometimes build actions. Tasks can be used to provide an easy way to include a complex set of functionality constructed from a number of different packages. By declaring dependencies on specific packages through a task, users of the task are freed of having to know these details themselves.

MontaVista Zone Content Server – Part of the MontaVista Zone, the Content Server is the location from which users locate and download all software content including MVL6 SDK, MSD(s), and any updates, security fixes and more.

Appendix A – Sample List of Test Procedures

Networking

- IPv4
- IPv6
- IPSEC
- IPSEC Hardware acceleration
- IP Tables
- VLAN
- SCTP
- PPP

Tools

- MVIP system build
- Linux Trace Toolkit
- PRAMFS
- KGDB over Ethernet
- KGDB over serial
- Multi-threaded Core Dump
- OProfile
- Powertop
- Kernel Function Tracing

I/O

- USB Host / HID
- USB Host / Ethernet
- USB Host / Mass Storage
- USB Host / Serial
- USB Full Speed
- USB High Speed
- Generic Keyboard & Mouse
- Real time clock
- Ethernet Networking
- Wireless Networking
- SD/MMC cards
- Other hardware specific tests per MSD test plan

General System

- Application spot checks
- System stress testing
- Linux Test Project
- Prelinking
- Memory Over commitment
- Out-of-Memory Killer

Power Management

- Basic power on/off control
- Dynamic tick
- Dynamic frequency and voltage scaling
- Deferrable Timers

Graphics and Sound

- Xorg
- DirectFB
- HW graphics acceleration
- ALSA

Real-time

- Preemption modes
- System stress under pre emption modes
- Priority Inheritance
- Robust Futexes
- Priority Queuing
- Completely Fair Scheduler (CFS)
- CPU Affinity
- High resolution timer

Application Development

- POSIX Test Suite
- POSIX message queues

Booting

- NFS root
- Disk
- Flash
- Bootloader

Filesystems

- EXT2/3
- JFFS2
- NFSv3
- FAT
- CRAMFS
- RAMFS
- SYSFS
- TMPFS
- Automount

Usage

- Installation testing
- Kernel build

Toolchain

- GCC Test Suite
- GDB Test Suite
- Binutils/Linker/Assembler Test Suite
- Plum Hall C/C++ Validation Suite
- GLIBC and libstdc++ Test Suite
- EEMBC Benchmarks

Appendix B – Sample Recipe

```
DESCRIPTION = “An Embeddable SQL Database Engine”
SECTION = “libs”
PRIORITY = “optional”
DEPENDS = “readline ncurses”
LICENSE = “PD”
SRC_URI = “http://www.sqlite.org/sqlite-${PV}.tar.gz \
                file://libtool.patch;patch=1”
S = “${WORKDIR}/sqlite-${PV}”
inherit autotools pkgconfig
EXTRA_OECONF = “--disable-tcl --enable-shared \
                --enable-threadsafe“
do_compile_prepend() {
                oe_runmake sqlite3.h
                install -m 0644 sqlite3.h ${STAGING_INCDIR}
}
do_stage() {
oe_libinstall -so libsqlite3 ${STAGING_LIBDIR}
install -m 0644 sqlite3.h ${STAGING_INCDIR}
}
PACKAGES = “libsqlite libsqlite-dev libsqlite-doc sqlite3 sqlite3-dbg”
FILES_sqlite3 = “${bindir}/*”
FILES_libsqlite = “${libdir}/*.so.*”
FILES_libsqlite-dev = “${libdir}/*.a ${libdir}/*.la ${libdir}/*.so \
                ${libdir}/pkgconfig ${includedir}”
FILES_libsqlite-doc = “${docdir} ${mandir} ${infodir}”
AUTO_LIBNAME_PKGS = “libsqlite”
```


MontaVista Software, Inc.
2929 Patrick Henry Drive
Santa Clara, CA 95054
Tel: +1.408.572.8000
Fax: +1.408.572.8005
email: sales@mvista.com
www.mvista.com

montavista™

montavista™