PATTON Software Release Strategy for Trinity™ OS

This paper covers Patton's strategy for managing the complex process of software development, maintaining discipline over the way software versions are created, updated, released and maintained.

PATTON®
let's Connect!™

Contents

Printed in the USA.

Introduction

Patton products comprise advanced hardware designs that run the company's proprietary Trinity™ software. Trinity comprises a Linux-based operating system (OS) combined with application software that supports various Patton hardware platforms. Continuous, ongoing software development is necessary for two main reasons:

- Address market requirements by adding features and functions that incorporate recent technological advances.
- Aggressively resolve and correct software "bugs" discovered in field deployments.

This paper covers Patton's strategy for managing the complex process of software development, maintaining discipline over the way software versions are created, updated, released and maintained.

Customer Profiles

Concerns, requirements and priorities vary from one customer to another when implementing a software upgrade. Some customers are aggressive, early adopters, eager to implement new technologies, features and functions. Others are more conservative, with a focus on preserving stable and reliable network service. Consider, for example, an Internet telephony service provider (ITSP) compared with an enterprise integrator.

An ITSP using Patton VoIP gateways is primarily concerned about system stability and backwards compatibility. Each software upgrade requires thorough testing in his network environment. The ITSP is reluctant to upgrade and will only do so in order to resolve software problems discovered in field deployment.

An enterprise system integrator, by contrast, is often willing to risk a system "hiccup" or two in order to implement the newest state-of-the-art technology and offer new services to his customer base.

So, when a software upgrade is required, one customer prefers maximum software stability, while the other seeks the newest technology to gain a competitive edge.

Patton's software development strategy provides scheduled new software releases and builds that address the needs of these two key customer profiles:

- **Aggressive customers**—for early adopters, provide the latest technology, features, and functions at regularly scheduled intervals
- **Conservative customers**—for risk-averse customers, provide consistently scheduled bug-fixes—at regularly scheduled intervals—with high stability and backwards compatibility

Terminology

Overview

The terms *release* and *build* appear throughout this document. A release may be described as a *major release* or a *minor release*, whereas a *build* is described as a *technology build* or a *maintenance build*. The following paragraphs define these terms and describe how they are related. Figure 1 on page 4 provides an overview of the software development process and terms, illustrating the following points:

- a major release contains several minor releases
- a release proceeds through technology and maintenance phases
- a release is instantiated by *builds* at regular intervals

Release

A *release* is an abstraction of a set of software features and functions designed to run on a hardware platform and published for customer use.

Major Release

A *major release* indicates a significant episode in the software development life-cycle. A major release usually involves a new or substantially revised software component, significant new functionality, or support for a new hardware platform. A major release tends to

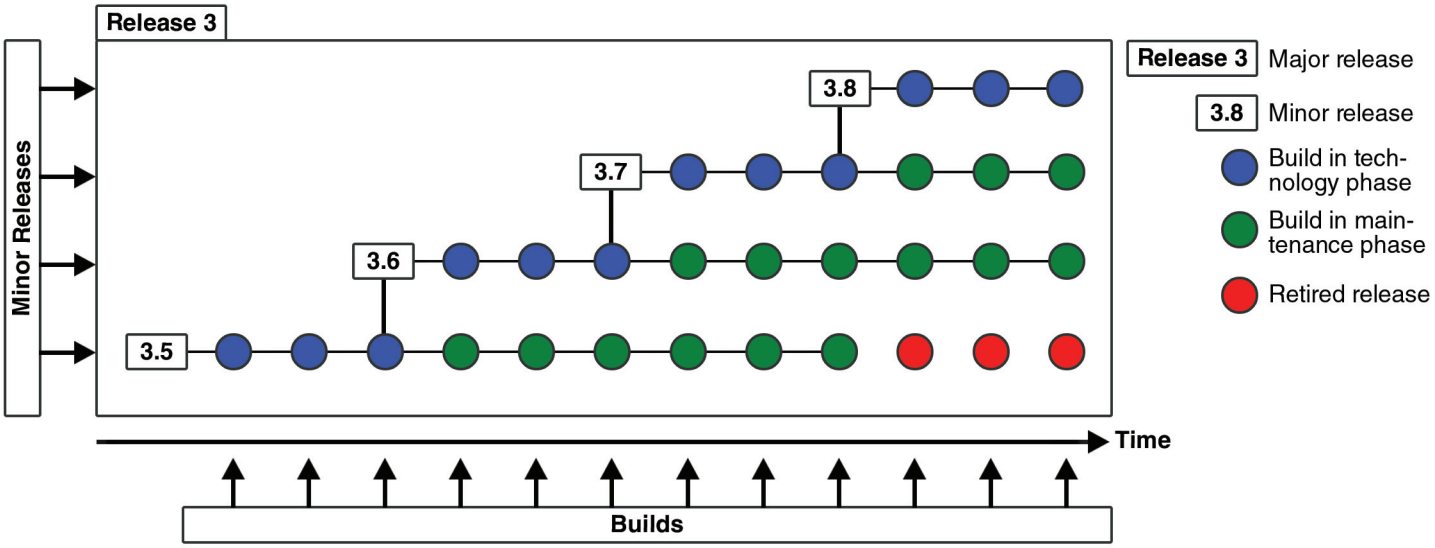


Figure 1 – Overview of software release terminology

involve increased risk when migrating from older releases. A major release typically comprises three *minor releases*. At any given point in time a major release will include one minor release in *technology phase* and two minor releases in *maintenance phase*. The first digit in the build descriptor indicates the major release.

Minor Release

A *minor release* indicates a step-forward in development (a revision) within a major release. The second digit in the build descriptor indicates a minor release.

Version

The third digit in the build descriptor indicates the software version within the major and minor release.

Build

A *build* is an instantiation (version) of a release at a fixed time within the development cycle. Each build consists of a set of software images that cover all supported products. The first three digits in the build descriptor together indicate the release version.

Technology Build

The purpose of a *technology build* is to deliver new technology, features, and functions that satisfy the requirements of aggressive, early-adopter customers. Patton creates technology builds for the first three instantiations of a minor release as it proceeds through a technology phase. A technology build will include all bug fixes and software corrections introduced to date. It may also add software support for upcoming new products.

Maintenance Build

The purpose of a *maintenance build* is to provide stable, field-proven software that provides bug fixes and backwards compatibility for products deployed in live network environments. The maintenance build delivers no new features that may potentially introduce new bugs. Any behavior changes in the software are restricted to those required by the bug fixes. A maintenance phase follows the technology phase of every minor release. The resulting maintenance build freezes (maintains) the behaviors and feature/function set of a minor release at a fixed point in time.

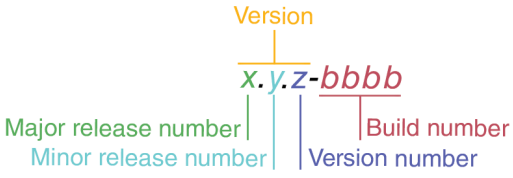


Figure 2. Software Build Descriptor

Anatomy of a Software Build Descriptor

Patton uses the software build descriptor (see figure 2) to manage and track software development. The descriptor has the format *x.y.z-bbbbb* where x indicates the major release number, *y* indicates the

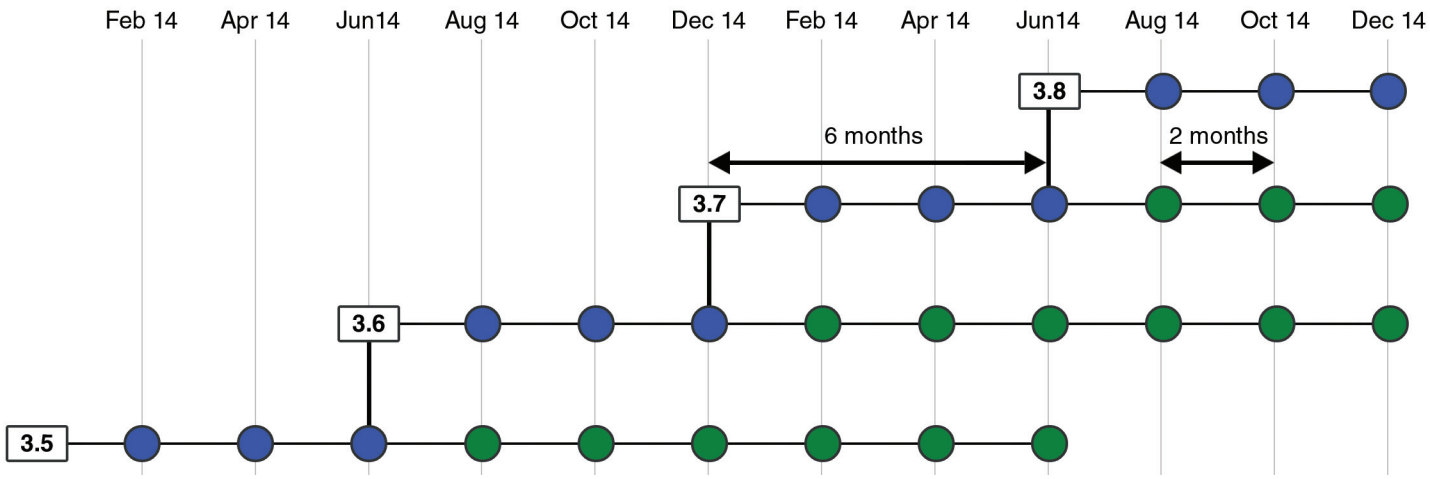


Figure 3. Trinity 3.x Release Roadmap

minor release number, and *x.y.z* together indicate the release version. The characters following the hyphen comprise the build number, which serves as a mechanism for tracking Patton's internal pre-release validation and fault correction processes.

Release Strategy

Overview

A new minor release is spawned twice per year (every six months) with builds published at two-month intervals. Figure 3 illustrates Patton's software development cycle. In this cycle, Patton publishes 9 builds for each release. The first three are technology builds and the last six are maintenance builds. Patton develops new features for incorporation into the technology build only. Bug fixes are incorporated into all builds. Patton supports three active releases at any given time, one build in technology phase and two builds in maintenance phase.

Software Development Phases

According to Patton's software release strategy, each software release passes through three development *phases* as described below:

1. **Technology phase (6 months)**—Builds released during the technology phase introduce new software features and functions.
2. **Maintenance phase (12 months)**—Only bug fixes are incorporated into new builds released during the maintenance phase.
3. **Retired (support-only) phase (6 months)**—No further builds are released. Patton Technical Services continues to support customer applications.

In order to make new software features accessible quickly and on a regular basis, the newest release always begins its lifecycle in the *technology phase*.

To provide stability and backwards compatibility, the two previous releases remain in the *maintenance phase*.

Example

As illustrated in figure 3, during August 2014 release 3.5 (the latest active technology release) spawns the new minor release 3.6. At that time release 3.6 becomes the new active release in technology phase, while release 3.5 transitions into maintenance phase. Release 3.6 will remain in technology phase for 6 months (until December 2014). Release 3.5 will remain in maintenance phase for the next 12 months, until June 2015, when it will transition into the support-only phase.

Summary

In summary, a new minor release is spawned twice per year (every six months), while builds for all active releases are created six times per year (every two months).

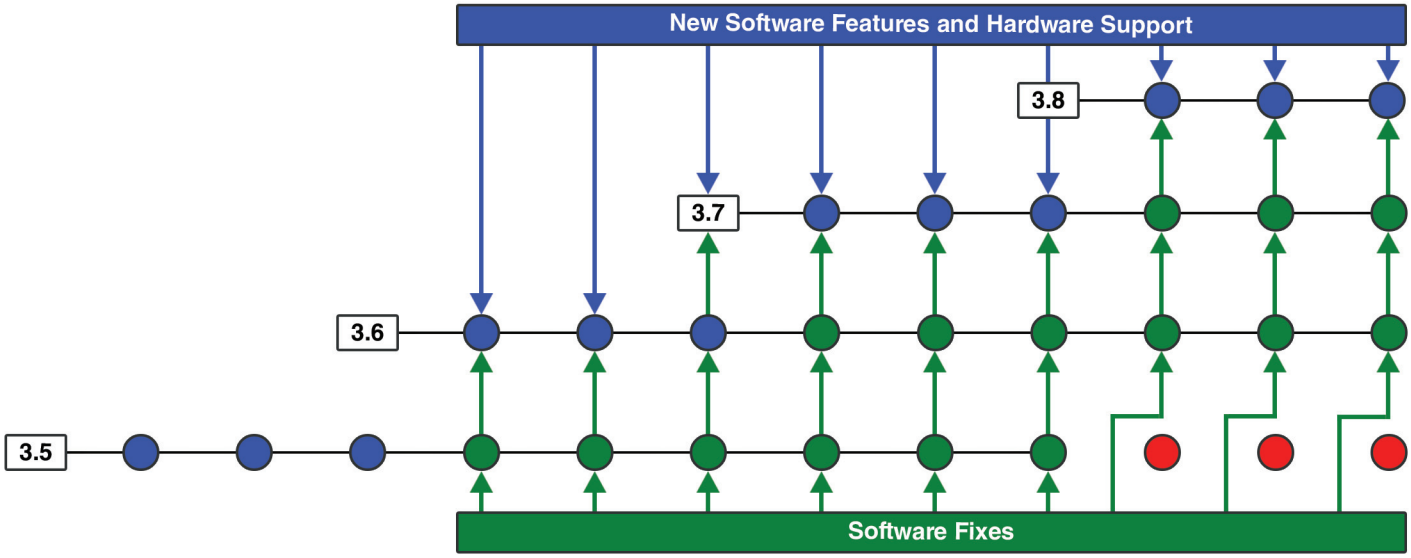


Figure 4. Software change synchronization

The plan outlined above provides rapid, regular access to new features in the technology builds with bug fixes in the subsequent maintenance builds.

Software Change Classification and Synchronization

When a bug fix is integrated into a maintenance build, it will be incorporated into builds for subsequent releases.

- **Bug fixes** are synchronized bottom-up. They are first fixed in the maintenance build of the oldest active release in which the bug is present, then subsequently propagated into all applicable newer releases, including the technology build.
- **New features** are synchronized top-down, incorporated only into the technology build (see figure 4).

Managing software changes

For each software change, the managing engineer must make a decision:

- **Is it a bug fix?**—integrate into **all** applicable releases
- **Is it a new feature?**—integrate **only** into the technology build.

Development schedule and discipline

Patton publishes new builds for all active minor releases 6 times a year (bi-monthly), in each even-numbered month. To accommodate software validation, all development stops at least 3 weeks before the scheduled release date. All builds are then thoroughly tested in Patton's validation lab. Before publishing the software builds Patton corrects any deficiencies discovered during validation.

All builds are tagged in the source repository, and built from a dedicated server. A strict build procedure is followed so that any build can be reproduced at a future date as required.

Release Life Cycle

Each minor release follows the following life cycle: after spawning from the previous release, the new release is first tested, then re-categorized from *technology* to *maintenance*, and finally to *retired (support-only)*. The phase of the release defines the availability and support provided for it.

Time Line (Months)	Milestone	State After Milestone	State Description
0	**FCS** First Commercial Shipment	**Technology**	New features are introduced and bugs are fixed. The release is publicly accessible to customers and supported. Builds are published every two months.
6	**EOD** End of Devlopment	**Maintenance**	All bug fixes are integrated. The release is publicly accessible to customers and supported. Builds are published every two months.
18	**EOE** End of Engineering	**Retired**	No engineering changes are introduced. The release is supported and Patton helps customers migrate to the next release. No new builds will be published. The latest build is still available for download.
24	**EOL** End of Life	**EOL**	The release is no longer supported. The latest build is still available for download but may be removed without prior notice.

Table 1 describes the states of a software release during its life cycle. Figure 5 illustrates the process.

Release lifecycle and concurrence

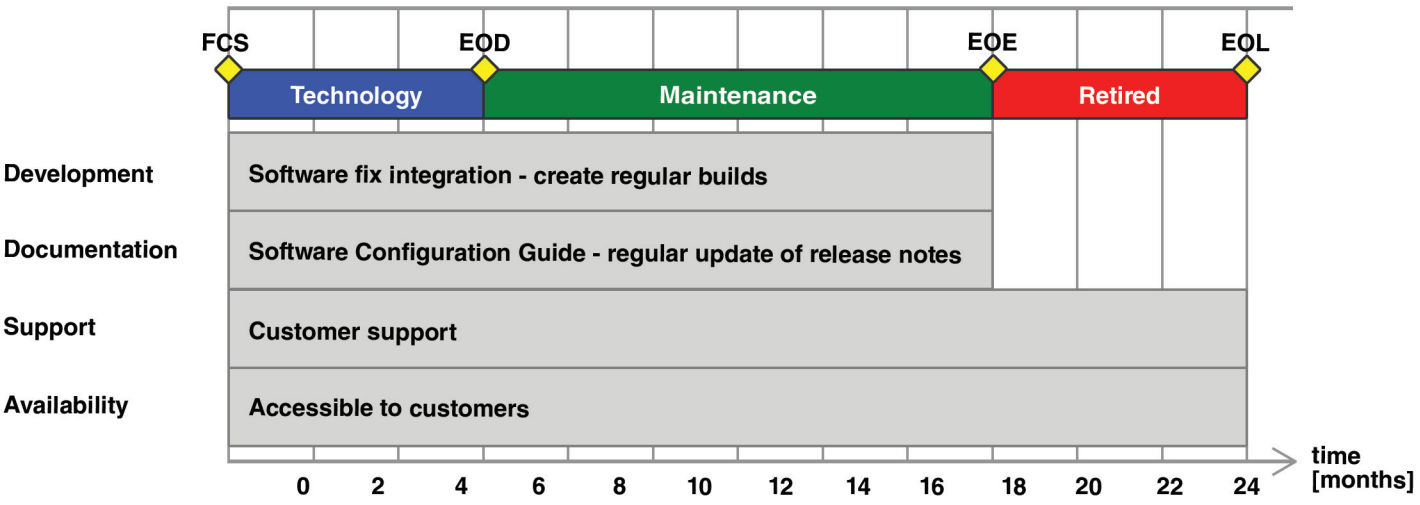
A release typically has a 24-month lifespan, with a new release spawned every 6 months. That means at any given time a maximum of 4 releases are concurrently available to customers: one in *technology phase*, two in *maintenance phase* and one in *retired (support-only) phase*.

Customer Deliverables

During the life cycle of a release, Patton provides the following deliverables to customers:

- Software Images
- Command Line Reference Guide
- Release Notes (RN)

The above deliverables are available from Patton's web site on upgrades.patton.com.



Legend: **FCS** – First Commercial Shipment | **EOD** – End of Development | **EOE** – End of Engineering | **EOL** – End of Life

Figure 5. Release life cycle

Software Images

A software image is part of a build. For each product, software images of the following builds are accessible:

- All technology and maintenance builds of a release before product End of Engineering (EOE).
- The latest maintenance build of each release in retired state

Older software images are usually available on request from Patton support:

- Email: support@patton.com
- Call: +1 301 975 1007

Command Line Reference Guide

The Command Line Reference Guide (software user manual) describes how to configure and operate the software of a specific release. Patton publishes one Command Line Reference Guide for each release.

Release Notes

Release Notes describe the software changes integrated since the previous build of the same release. Release Notes are published together with each software image of a new build. For each build the Release Notes document the history of accumulated changes since the previous build of the same release.

Conclusion

Patton's strategy for software upgrades and releases distinguishes technology builds from maintenance builds to achieve the objectives set forth in Customer Profiles on page 3. Patton makes new software features available quickly to early adopters while maintaining stability and backwards compatibility for more conservative customers.

The current technology build delivers the newest software features, while two maintenance builds provide bug fixes while preserving stable, backwards-compatible software functionality. For each major release, one minor release in technology phase and two minor releases in maintenance phase are engineered and supported concurrently. For each release, updates are published regularly in the form of builds for all hardware platforms the release supported.

The advantages of this strategy are:

- Stability and quality
  - Software upgrades are published on a regular basis.

    Features are frozen during the entire maintenance phase of a release.
  - Only minor changes, mostly bug fixes, are made within a maintenance build.
  - All builds created in the maintenance phase are fully backwards compatible.
- Faster time-to-market for new features
  - New features are integrated into technology builds which are published at regular intervals
  - More room for adaptations and customer acceptance.
  - Accurate documentation—the associated Command Line Reference Guide completely describes the software functionality contained in a release.
  - Release Notes document the history of changes between updates

PATTON®
let's Connect!™
7622 Rickenbacker Drive
Gaithersburg, MD 20879 USA
tel: **+1.301.975.1007**
fax: **+1.301.869.9293**
web: **www.patton.com**
email: **marketing@patton.com**
Document: 07M-TRINITY-REL-WP2