



Red Hat Satellite 5.6

Satellite Errata Management

Edition 3

Rich Jerrido

Rich Jerrido

Red Hat Satellite 5.6 Satellite Errata Management

Edition 3

Rich Jerrido
Red Hat Solutions Architects
rwj@redhat.com

Legal Notice

Copyright 2013 | Red Hat, Inc. |. This document is licensed by Red Hat under the Creative Commons Attribution-ShareAlike 3.0 Unported License. If you distribute this document, or a modified version of it, you must provide attribution to Red Hat, Inc. and provide a link to the original. If the document is modified, all Red Hat trademarks must be removed. Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law. Red Hat, Red Hat Enterprise Linux, the Shadowman logo, JBoss, MetaMatrix, Fedora, the Infinity Logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries. Linux is the registered trademark of Linus Torvalds in the United States and other countries. Java is a registered trademark of Oracle and/or its affiliates. XFS is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries. MySQL is a registered trademark of MySQL AB in the United States, the European Union and other countries. Node.js is an official trademark of Joyent. Red Hat Software Collections is not formally related to or endorsed by the official Joyent Node.js open source or commercial project. The OpenStack Word Mark and OpenStack Logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community. All other trademarks are the property of their respective owners.

Keywords

Abstract

Strategies and techniques useful for implementing an errata management framework using Red Hat Satellite

Table of Contents

Preface	3
1. Document Conventions	3
1.1. Typographic Conventions	3
1.2. Pull-quote Conventions	4
1.3. Notes and Warnings	5
2. Feedback!	5
Overview	6
1. Introduction	6
2. Audience	6
3. Requirements	6
4. Goals	6
Chapter 1. Policy Considerations & Definitions	7
1.1. Policy Considerations	7
1.2. Change Types	7
1.2.1. Standard Changes	7
1.2.2. Normal Changes	8
1.2.3. Emergency Changes	8
Chapter 2. Configuration Setup	9
2.1. Updating the Satellite	9
2.2. Clone Channel Refresher	9
2.3. Implementation Overview	9
2.4. Creating the Configuration for spacewalk-clone-by-date	10
Chapter 3. Monthly Patching Example	12
3.1. Standard Changes	12
3.2. Normal & Emergency Changes	13
Chapter 4. Quarterly Patching Example	14
4.1. Configuration File Changes	14
4.2. Standard Changes	14
4.3. Normal & Emergency Changes	16
Chapter 5. Adapting This Document	17
Chapter 6. Example Patch Policy	18
Using the mergeErrata.py script	19
Extended Update Support (EUS)	22
New Features of spacewalk-clone-by-date in Red Hat Satellite 5.6	24
Additional Reading & References	25
Revision History	26

Preface

1. Document Conventions

This manual uses several conventions to highlight certain words and phrases and draw attention to specific pieces of information.

In PDF and paper editions, this manual uses typefaces drawn from the [Liberation Fonts](#) set. The Liberation Fonts set is also used in HTML editions if the set is installed on your system. If not, alternative but equivalent typefaces are displayed. Note: Red Hat Enterprise Linux 5 and later include the Liberation Fonts set by default.

1.1. Typographic Conventions

Four typographic conventions are used to call attention to specific words and phrases. These conventions, and the circumstances they apply to, are as follows.

Mono-spaced Bold

Used to highlight system input, including shell commands, file names and paths. Also used to highlight keys and key combinations. For example:

To see the contents of the file **my_next_bestselling_novel** in your current working directory, enter the **cat my_next_bestselling_novel** command at the shell prompt and press **Enter** to execute the command.

The above includes a file name, a shell command and a key, all presented in mono-spaced bold and all distinguishable thanks to context.

Key combinations can be distinguished from an individual key by the plus sign that connects each part of a key combination. For example:

Press **Enter** to execute the command.

Press **Ctrl+Alt+F2** to switch to a virtual terminal.

The first example highlights a particular key to press. The second example highlights a key combination: a set of three keys pressed simultaneously.

If source code is discussed, class names, methods, functions, variable names and returned values mentioned within a paragraph will be presented as above, in **mono-spaced bold**. For example:

File-related classes include **filesystem** for file systems, **file** for files, and **dir** for directories. Each class has its own associated set of permissions.

Proportional Bold

This denotes words or phrases encountered on a system, including application names; dialog box text; labeled buttons; check-box and radio button labels; menu titles and sub-menu titles. For example:

Choose **System** → **Preferences** → **Mouse** from the main menu bar to launch **Mouse Preferences**. In the **Buttons** tab, select the **Left-handed mouse** check box and click **Close** to switch the primary mouse button from the left to the right (making the mouse suitable for use in the left hand).

To insert a special character into a **gedit** file, choose **Applications** → **Accessories** → **Character Map** from the main menu bar. Next, choose **Search** → **Find...** from the **Character Map** menu bar, type the name of the character in the **Search** field and click **Next**. The character you sought will be highlighted in the **Character Table**. Double-click this highlighted character to place it in the **Text to copy** field and then click the **Copy** button. Now switch back to your document and choose **Edit** → **Paste** from the **gedit** menu

bar.

The above text includes application names; system-wide menu names and items; application-specific menu names; and buttons and text found within a GUI interface, all presented in proportional bold and all distinguishable by context.

Mono-spaced Bold Italic* or *Proportional Bold Italic

Whether mono-spaced bold or proportional bold, the addition of italics indicates replaceable or variable text. Italics denotes text you do not input literally or displayed text that changes depending on circumstance. For example:

To connect to a remote machine using ssh, type **ssh *username@domain.name*** at a shell prompt. If the remote machine is **example.com** and your username on that machine is john, type **ssh john@example.com**.

The **mount -o remount *file-system*** command remounts the named file system. For example, to remount the **/home** file system, the command is **mount -o remount /home**.

To see the version of a currently installed package, use the **rpm -q *package*** command. It will return a result as follows: ***package-version-release***.

Note the words in bold italics above — username, domain.name, file-system, package, version and release. Each word is a placeholder, either for text you enter when issuing a command or for text displayed by the system.

Aside from standard usage for presenting the title of a work, italics denotes the first use of a new and important term. For example:

Publican is a *DocBook* publishing system.

1.2. Pull-quote Conventions

Terminal output and source code listings are set off visually from the surrounding text.

Output sent to a terminal is set in **mono-spaced roman** and presented thus:

```
books      Desktop  documentation  drafts  mss    photos  stuff  svn
books_tests Desktop1  downloads      images  notes  scripts svgs
```

Source-code listings are also set in **mono-spaced roman** but add syntax highlighting as follows:


```
static int kvm_vm_ioctl_deassign_device(struct kvm *kvm,
                                       struct kvm_assigned_pci_dev *assigned_dev)
{
    int r = 0;
    struct kvm_assigned_dev_kernel *match;

    mutex_lock(&kvm->lock);

    match = kvm_find_assigned_dev(&kvm->arch.assigned_dev_head,
                                  assigned_dev->assigned_dev_id);
    if (!match) {
        printk(KERN_INFO "%s: device hasn't been assigned before, "
                       "so cannot be deassigned\n", __func__);
        r = -EINVAL;
        goto out;
    }

    kvm_deassign_device(kvm, match);

    kvm_free_assigned_device(kvm, match);

out:
    mutex_unlock(&kvm->lock);
    return r;
}
```

1.3. Notes and Warnings

Finally, we use three visual styles to draw attention to information that might otherwise be overlooked.



Note

Notes are tips, shortcuts or alternative approaches to the task at hand. Ignoring a note should have no negative consequences, but you might miss out on a trick that makes your life easier.



Important

Important boxes detail things that are easily missed: configuration changes that only apply to the current session, or services that need restarting before an update will apply. Ignoring a box labeled 'Important' will not cause data loss but may cause irritation and frustration.



Warning

Warnings should not be ignored. Ignoring warnings will most likely cause data loss.

2. Feedback!

If you find a typographical error in this manual, or if you have thought of a way to make this manual better, we would love to hear from you! Please contact the author of this guide, Rich Jerrido (rwj@redhat.com). If you have a suggestion for improving the documentation, try to be as specific as possible when describing it. If you have found an error, please include the section number and some of the surrounding text so we can find it easily.

Overview

1. Introduction

This document is designed to provide guidance as to how a user of Red Hat Satellite can implement an errata management solution to support a rigid lifecycle for their Red Hat Enterprise Linux systems. This document is designed to bridge the gap between an Organization's patching methodology and the technical implementation to support that methodology.

2. Audience

This guide is intended to be used by Systems & Security Administrators responsible for implementing a scalable and rigid patching methodology.

3. Requirements

This guide assumes the reader is familiar with Red Hat Satellite and its terminology.

Table 1. Additional Requirements

Requirement	Notes
Red Hat Satellite	This guide depends heavily on the spacewalk-clone-by-date command, introduced with Red Hat Satellite 5.4 via RHEA-2012-0330 . The examples in this document use features of spacewalk-clone-by-date,(such as the ability to clone only security errata) introduced with Red Hat Satellite 5.5. It is expected that the user is running Red Hat Satellite 5.4 at a minimum, and preferably Red HatSatellite 5.5 (or greater) to get optimal usage from this guide.
Permissions	It is expected that the user account used for the examples has been granted either the Organization Administrator or Channel Administrator role.

4. Goals

This document has the following goals:

- ▶ To build a flexible framework for deploying Red Hat Enterprise Linux errata
- ▶ To ensure that errata is properly progressed from environment to environment (Dev->QA->Prod)
- ▶ To decouple the release of errata from Red Hat from the release of errata within one's environment
- ▶ To provide a means for an organization to remain fairly current with the latest errata, without being too current

Chapter 1. Policy Considerations & Definitions

1.1. Policy Considerations

Firstly, before implementing a technical solution, it is advisable to take a look at the many concerns that ultimately influence a patching strategy. However, regardless of the considerations that an organization may have, the framework that is being assembled will remain the same. Listed below are some considerations that an organization may consider when designing a patching strategy.

Table 1.1. Considerations

Concerns	Definitions
Conservativeness of patch frequency	How aggressive or conservative the organization is with regards to how often they patch
Conservativeness of patch type	Which errata types (RHBA RHSA RHEA) is the organization willing to deploy
Scheduling of resources	Many organizations have post change validation tasks of varying levels of rigidity that must occur after patching. This also influences the policy
Bake-in time	The amount of time between patching each environment which would time to identify any issues or regressions.
Regulatory Compliance	The requirements of many regulations and compliance standards such as PCI, SOX, & HIPAA mandate that errata must be applied within a particular timeframe. As an example, the Payment Card Industry Data Security Standard (PCI DSS) states: "Ensure that all system components and software are protected from known vulnerabilities by having the latest vendor-supplied security patches installed. Install critical security patches within one month of release". This would influence an organization to adopt a more frequent patching methodology for their systems.

1.2. Change Types

It is useful to align the deployment of errata to Red Hat Enterprise Linux systems based upon the level of change and impact that it will have on the system in question. ITIL (®) , defines three types of changes within the scope of Change Management. These three types of changes should cover most scenarios. This guide will focus on delivering a methodology that fits into this framework.

1.2.1. Standard Changes

A standard change is a change to a service or infrastructure for which the approach is pre-authorised by change management that has an accepted and established procedure to provide a specific change requirement. The elements of a standard change are:

- There is a defined trigger to initiate the request for change
- The activities/tasks are well known, documented and proven
- Authority is given in advance (these changes are pre-authorised)
- The risk is usually low

Most patching of Red Hat Enterprise Linux systems fall into the category of Standard Changes. Standard Changes can (and should) be automated as they do not require a reboot.

1.2.2. Normal Changes

A normal change refers to changes that must follow the complete change management process. Normal changes are often categorised according to risk and impact to the organisation/business. For example, minor change – low risk and impact, significant change – medium risk and impact and major change – high risk and impact.

- Record requests for change
- Assess and evaluate the change
- Change planning and scheduling

Some packages, such as the kernel, glibc, and potentially 3rd party software may require additional human interaction due to either a required reboot, coordination around downtime periods, or some form of validation. Additionally, as updates to the kernel require a reboot and associated outage, many organizations choose to handle other updates, such as hardware maintenance or firmware updates at this time.

1.2.3. Emergency Changes

Emergency change is reserved for changes intended to repair an error in an IT service that is impacting the business to a high degree or to protect the organisation from a threat. Emergency Changes include:

- Critical zero-day vulnerabilities that must be addressed prior to the next patch cycle.
- Bugfixes for production issues that must be addressed prior to the next patch cycle

Chapter 2. Configuration Setup

The configuration files created below will be used for each example with minor modifications.

2.1. Updating the Satellite

The Red Hat Satellite should be kept up to date with the latest errata available on Red Hat Network. The `satellite-sync` command should be run fairly often, preferably daily. This allows the Red Hat Satellite to have the most up to date errata available, and within the organization's network. This process can be automated using the process described in the [Red hat Satellite Installation Guide](#). It is expected that systems registered to the Red Hat Satellite are NOT registered to the Red Hat provided channels (as their content will change daily), but to the clone channels that will be created below.

2.2. Clone Channel Refresher

Clone channels within Red Hat Satellite provide a means to take a snapshot of a particular channel and populate it with only the errata that is needed. This provides the ability to prune the errata list that is made available to systems. It also provides a means to decouple the process of downloading new errata and packages from Red Hat Network (via the `satellite-sync` command), from the process of making it available to the systems registered to the Satellite. More information on clone channels is provided in the [Red Hat Satellite Getting Started Guide](#).

2.3. Implementation Overview

We will create three cloned parent channels on Red Hat Satellite to reflect the Dev/QA/Prod environments. Using `spacewalk-clone-by-date`, errata will be released to the cloned channels in a staged manner, ensuring that content is made available for Dev, then QA, then Prod. For the purposes of this document, the RHEL6 x86_64 RHN channel (`rhel-x86_64-server-6`) will be used as a parent channel, and the RHEL6 x86_64 RHN tools channel (`rhn-tools-rhel-x86_64-server-6`) as a child channel. This can be adapted for any parent channel, including those provided by the Extended Update Support (EUS) and Extended Lifecycle Support (ELS) add-ons. See [Appendix B, Extended Update Support \(EUS\)](#) for more information on EUS. Also, additional child channels, which reflect other products can be added. Additionally, at the end of this procedure, there will be clone channels created for each of the Dev/QA/Prod environments with the following labels:

Table 2.1. Channel Mappings

Channel Label	Purpose
<code>rhel-x86_64-server-6</code>	RHEL6 Server RPMs
<code>rhn-tools-rhel-x86_64-server-6</code>	RHN Tools RPMs
<code>dev-rhel-x86_64-server-6</code>	Clone of RHEL6 Server RPMs for the Development Environment
<code>dev-rhn-tools-rhel-x86_64-server-6</code>	Clone of RHEL6 RHN Tools RPMs for the Development Environment
<code>qa-rhel-x86_64-server-6</code>	Clone of RHEL6 Server RPMs for the QA Environment
<code>qa-rhn-tools-rhel-x86_64-server-6</code>	Clone of RHEL6 RHN Tools RPMs for the QA Environment
<code>prod-rhel-x86_64-server-6</code>	Clone of RHEL6 Server RPMs for the Production Environment
<code>prod-rhn-tools-rhel-x86_64-server-6</code>	Clone of RHEL6 RHN Tools RPMs for the Production Environment

It is expected that the various systems registered with Red Hat Satellite will be assigned to the {dev,qa,prod} channels based upon them being Development, QA & Production systems respectively. It is also expected that new systems will be assigned to which environment they belong to. For existing

systems, this can be done via the System Set Manager. For new systems, it is preferable to use an Activation Key to accomplish this task.

2.4. Creating the Configuration for spacewalk-clone-by-date

The following configuration files will be used to allow spacewalk-clone-by-date to understand the mapping between a source channel and its respective clone channel, as well as allowing the end user to modify the behavior of the tool to further satisfy their business requirements.

Procedure 2.1.

1. On the Satellite Server, create a file (dev-channels.conf) containing the following content:

```
{
  "username": "cadmin",
  "password": "*****",
  "assumeyes": true,
  "skip_depsolve": false,
  "security_only": false,
  "blacklist": {
    "ALL": [
      "kernel*"
    ]
  },
  "removelist": {
    "ALL": [
      ""
    ]
  },
  "channels": {
    "rhel-x86_64-server-6": "dev-rhel6-x86_64-server-6",
    "rhn-tools-rhel-x86_64-server-6": "dev-rhn-tools-rhel-x86_64-server-6"
  }
}
```

2. Next, on the Satellite Server, create a file (qa-channels.conf) containing the following content:

```
{
  "username": "cadmin",
  "password": "*****",
  "assumeyes": true,
  "skip_depsolve": false,
  "security_only": false,
  "blacklist": {
    "ALL": [
      "kernel*"
    ]
  },
  "removelist": {
    "ALL": [
      ""
    ]
  },
  "channels": {
    "rhel-x86_64-server-6": "qa-rhel6-x86_64-server-6",
    "rhn-tools-rhel-x86_64-server-6": "qa-rhn-tools-rhel-x86_64-server-6"
  }
}
```

3. Finally, on the Satellite Server, create a file (prod-channels.conf) containing the following content:

```
{
  "username": "cadmin",
  "password": "*****",
  "assumeyes": true,
  "skip_depsolve": false,
  "security_only": false,
  "blacklist": {
    "ALL": [
      "kernel*"
    ]
  },
  "removelist": {
    "ALL": [
      ""
    ]
  },
  "channels": {
    "rhel-x86_64-server-6": "prod-rhel6-x86_64-server-6",
    "rhn-tools-rhel-x86_64-server-6": "prod-rhn-tools-rhel-x86_64-server-6"
  }
}
```

The configuration files are in JavaScript Object Notation (JSON) format, which makes it easy to parse this information, either by humans or programmatically. Additionally, these options can be overridden on the command line. The configuration options are as follows (and can be referenced in the spacewalk-clone-by-date manual):

Table 2.2. spacewalk-clone-by-date options

Option	Meaning
username	The user whom spacewalk-clone-by-date will connect as
password	The password of the above user. Note: if this entry is omitted, spacewalk-clone-by-date will prompt for it
assumeeyes	Tells spacewalk-clone-by-date to not ask for confirmation
to_date	Clone errata up to the specified date, specified in YYYY-MM-DD format. Our examples do not use this parameter as the errata cloning date will be explicitly specified on the command line"
skip_depsolve	Disable dependency checking (Not Recommended)
security_only	Clone only security errata
blacklist	List of packages to be removed after cloning.
removelist	List of packages to be removed after cloning, including those not added by errata/channel cloning
channels	The mapping between source and destination channels

Chapter 3. Monthly Patching Example

In this example, a fictitious company wishes to deploy errata to their systems with the following criteria:

- Each month's 'patch bundle' will include all errata up to the 1st day of that month. That is, in February systems will be patched with all errata prior to 1 Feb.
- Development shall be patched on the 1st Saturday of the month.
- QA shall be patched on the 2nd Saturday of the month.
- Production shall be patched on the 4th Saturday of the month.
- They want to deploy all relevant errata (Security [RHSA], Bugfix [RHBA], & Enhancements [RHEA]) to their systems with the exception of kernel errata.
- Quarterly, they wish to deploy kernel errata to coincide with a scheduled outage window.

For the purpose of this section, the following calendar will be used to refer to dates:

January 2013							February 2013							March 2013							
Su	Mo	Tu	We	Th	Fr	Sa	Su	Mo	Tu	We	Th	Fr	Sa	Su	Mo	Tu	We	Th	Fr	Sa	
		1	2	3	4	5							1	2						1	2
6	7	8	9	10	11	12	3	4	5	6	7	8	9	3	4	5	6	7	8	9	
13	14	15	16	17	18	19	10	11	12	13	14	15	16	10	11	12	13	14	15	16	
20	21	22	23	24	25	26	17	18	19	20	21	22	23	17	18	19	20	21	22	23	
27	28	29	30	31			24	25	26	27	28			24	25	26	27	28	29	30	
														31							

3.1. Standard Changes

Procedure 3.1.

1. On the first Saturday of the month (2 Feb), the dev-rhel-x86_64-server-6 channel will be populated with errata released prior to 1 Feb.

```
spacewalk-clone-by-date --config=dev-channels.conf -d 2013-02-01
```

2. At this point, the content of the dev-rhel-x86_64-server-6 channel and its children have been 'fast-forwarded' to 1 Feb as shown below.

	Feb	March	April
Dev	--- --->		
QA			
Prod			

3. Next, and also on the first Saturday of the month (2 Feb), all systems subscribed to the dev-rhel-x86_64-server-6 channel will be patched.
4. On the 2nd Saturday of February (9 Feb), spacewalk-clone-by-date is run again, this time specifying the configuration file for the QA environment.

```
spacewalk-clone-by-date --config=qa-channels.conf -d 2013-02-01
```

5. Next, all systems subscribed to the qa-rhel-x86_64-server-6 channel will be patched. Looking at the channel diagram again:

	Feb	March	April
Dev	--- --->		
QA	--- --->		
Prod			

6. Lastly, on the 4th Saturday of the February (23 Feb), spacewalk-clone-by-date is run again, specifying the configuration file for the PROD environment.

```
spacewalk-clone-by-date --config=prod-channels.conf -d 2013-02-01
```

7. Next, all systems subscribed to the prod-rhel-x86_64-server-6 channel will be patched. Looking at the channel diagram once more:

	Feb	March	April
Dev	--- --->		
QA	--- --->		
Prod	--- --->		

8. The next month, on the 1st Saturday (2 Mar), this process will be repeated, substituting the new date (2013-03-01) to the spacewalk-clone-by-date tool.

	Feb	March	April
Dev	--- --->--- --->		
QA	--- --->		
Prod	--- --->		

3.2. Normal & Emergency Changes

As per the criteria for our patching policy, kernel errata needs to be made available quarterly so that the systems may install it. This can be done via the UI using the process listed in the [Red Hat Satellite Getting Started Guide](#). This process will also be used for any errata that needs to be applied prior to the next patch cycle. It is expected that the administrator clones the errata in a similar manner as above, where it is cloned to Dev first, then QA, then Prod.

Alternatively, one may wish to clone these one-off errata non-interactively. Provided in Appendix A is an example script (mergeErrata.py) to perform merging of errata from one channel to another.

Chapter 4. Quarterly Patching Example

In this example, our fictitious company wishes to be a bit more conservative. They wish to deploy errata to their systems quarterly and with the following criteria:

- Each quarter's 'patch bundle' will include all errata up to the 1st day of that quarter. That is, for Q1, systems will be patched with all errata prior to 1 Jan.
- Development shall be patched on the 1st day of the first month of the quarter.
- QA shall be patched on the 1st day of the second month of the quarter.
- Production shall be patched on the 1st day of the third month of the quarter.
- They want to deploy all only security errata (RHSA) to their systems, with the exception of kernel errata.
- Quarterly, they wish to deploy kernel errata to coincide with a scheduled outage window.

For the purpose of this section, the following calendar will be used to refer to dates:

January 2013							February 2013							March 2013							
Su	Mo	Tu	We	Th	Fr	Sa	Su	Mo	Tu	We	Th	Fr	Sa	Su	Mo	Tu	We	Th	Fr	Sa	
		1	2	3	4	5							1	2						1	2
6	7	8	9	10	11	12	3	4	5	6	7	8	9	3	4	5	6	7	8	9	
13	14	15	16	17	18	19	10	11	12	13	14	15	16	10	11	12	13	14	15	16	
20	21	22	23	24	25	26	17	18	19	20	21	22	23	17	18	19	20	21	22	23	
27	28	29	30	31			24	25	26	27	28			24	25	26	27	28	29	30	
														31							

4.1. Configuration File Changes

To support the requirement to deploy only security errata, the spacewalk-clone-by-date configuration files ({dev,qa,prod}-channels.conf) need to be updated. Change:

```
"security_only": false,
```

to

```
"security_only": true,
```



Important

The version of spacewalk-clone-by-date in Red Hat Satellite 5.4 does not include support for cloning security errata. Only the version included in Red Hat Satellite 5.5 and above includes this functionality. This procedure will not work on Red Hat Satellite 5.4 without modifications.



Note

The way that errata are packaged means that some security errata contain bugfixes and enhancements. Example is [RHSA-2012:1269](#) which is an update for the qpid package. It contains a bugfix, security fix, and enhancement within one RPM and would be counted as a 'security errata' and would be cloned as security errata. It is important to note that the 'security_only' directive delivers "only errata which contain a security erratum" and not "security errata, and nothing more".

4.2. Standard Changes

Procedure 4.1.

1. On the first day of the first month of the quarter (1 Jan), the dev-rhel-x86_64-server-6 channel will be populated with errata released prior to 1 Jan:

```
spacewalk-clone-by-date --config=dev-channels.conf -d 2013-01-01
```

2. At this point, the content of the dev-rhel-x86_64-server-6 channel and its children have been 'fast-forwarded' to 1 Jan as shown below.

	Q1	Q2	Q3	
Dev	--- --->			Jan Apr Jul
QA				Feb May Aug
Prod				Mar Jun Sep

3. Next, and also on the first day of the first month of the quarter (1 Jan), all systems subscribed to the dev-rhel-x86_64-server-6 channel will be patched.
4. On the 1st day of the second month of the quarter (1 Feb), spacewalk-clone-by-date is run again, this time specifying the configuration file for the QA environment:

```
spacewalk-clone-by-date --config=qa-channels.conf -d 2013-01-01
```

5. Next, all systems subscribed to the qa-rhel-x86_64-server-6 channel will be patched. Looking at the channel diagram again:

	Q1	Q2	Q3	
Dev	--- --->			Jan Apr Jul
QA	--- --->			Feb May Aug
Prod				Mar Jun Sep

6. Lastly, on the 1st day of the third month of the quarter (1 Mar), spacewalk-clone-by-date is run again, specifying the configuration file for the PROD environment:

```
spacewalk-clone-by-date --config=prod-channels.conf -d 2013-01-01
```

7. Next, all systems subscribed to the prod-rhel-x86_64-server-6 channel will be patched. Looking at the channel diagram once more:

	Q1	Q2	Q3	
Dev	--- --->			Jan Apr Jul
QA	--- --->			Feb May Aug
Prod	--- --->			Mar Jun Sep

8. The next quarter, on the 1st day of the 1st month of the quarter (1 Apr), this process will be repeated, substituting the new date (2013-04-01) to the spacewalk-clone-by-date tool.

	Q1	Q2	Q3	
Dev	--- --->---	--->		Jan Apr Jul
QA	--- --->			Feb May Aug
Prod	--- --->			Mar Jun Sep

4.3. Normal & Emergency Changes

As per the criteria for our patching policy, kernel errata needs to be made available quarterly so that the systems may install it. This can be done via the UI using the process listed in the [Red Hat Satellite Getting Started Guide](#). This process will also be used for any errata that needs to be applied prior to the next patch cycle. It is expected that the administrator clones the errata in a similar manner as above, where it is cloned to Dev first, then QA, then Prod.

Alternatively, one may wish to clone these one-off errata non-interactively. Provided in Appendix A is an example script (mergeErrata.py) to perform merging of errata from one channel to another.

Chapter 5. Adapting This Document

In the previous chapters, examples were given as to how the spacewalk-clone-by-date tool can be used to support a structured errata management program. The examples used a relatively simple organization which only has a Development, QA and Production environment. Many organizations may want to apply differing policies to systems in different security zones, such as internal systems versus internet facing systems. This document can be customized in the following manner:

Table 5.1. Example Customizations

Customization	How to implement
Placing a subset of systems on a differing errata policy	Create new configuration files (and channels) for those systems
Deploying only security errata	change the 'security_only' directive to 'true'
Blacklist (prune) the errata list to remove unneeded packages	change the 'blacklist' directive to reflect the packages that aren't needed
Make the time between errata deployments more/less conservative	change the time between each run of spacewalk-clone-by-date to meet your requirements
Add additional products (child channels)	update the configuration files to add new source & destination channels

Chapter 6. Example Patch Policy

While this document focuses on the technical implementation of a patching methodology, many organizations also require a written patch policy. Listed below is a framework for a patching policy that can be adapted as needed.



Important

The below is provided as-is, without warranty, and does not constitute any legal advice of any variety

Table 6.1. Example Patch Policy

LINUX-INF-SEC-01	
Policy	COMPANY requires that all Linux Systems on the COMPANY network shall adhere to a pre-determined Patching Policy
Reason	To maintain a standardized support model for all devices connected to the network, devices must have a set of minimum requirements that will ensure the safety and integrity of all data and manageable systems. Additionally, auditory compliance requirements dictate that all systems be patched with vendor supplied patches in a timely fashion.
Background	Current standards for the DEPARTMENT team are to manage the hardware, operating system and ensure the system is maintained by all current security fixes/patches. Unpatched systems may expose the network to unforeseen events that may cause harm to the integrity of COMPANY networked environment.
Scope	<p>The scope of this document is limited to the following server operating systems as managed by the DEPARTMENT team:</p> <ul style="list-style-type: none"> ▸ Red Hat Enterprise Linux (RHEL 5.x / 6.x) ▸ Variants: Server / Workstation / Client / ComputeNode
Guidelines	<p>All security & bugfix errata will be applied to COMPANY Linux/Unix Servers under the following guidelines:</p> <ul style="list-style-type: none"> ▸ Notification of all patch deployment to business owners will occur no later than 3 business days prior to deployment. ▸ Development systems will be patched on the 1st Saturday of every month ▸ QA systems will be patched on the 2nd Saturday of every month ▸ Production systems will be patched on the 4th Saturday of every month ▸ Kernel upgrades will normally be applied semi-annually via separate change records ▸ DEPARTMENT reserves the right to deploy emergency errata necessary for critical security & bug fixes via unscheduled change and downtime windows. <p>DEPARTMENT reserves the right to deny network access to any system that does not conform to standard patching and support policies, and also assume control of any system until it has attained compliance with this policy.</p>
Conditions of Use	Only members of the DEPARTMENT team are authorized to make modifications to the architecture or configuration settings established within this policy.
Exceptions	NONE

Using the mergeErrata.py script

Provided below is a sample script (mergeErrata.py) that can be used to merge selected errata from one clone channel to another. Usage:

```
./mergeErrata.py -l admin -p **** -s satellite.domain.com -f rhel-x86_64-server-6 -d dev-rhel6-x86_64-server-6 -e RHSA-2013:0737,RHBA-2013:0735
```

Help:

```
./mergeErrata.py -h
Usage: mergeErrata.py [options]

Options:
  -h, --help                show this help message and exit
  -l LOGIN, --login=LOGIN    Login user for Red Hat Satellite/Hosted
  -p PASSWORD, --password=PASSWORD
                              Password for specified user. Will prompt if omitted
  -s SERVERFQDN, --server=SERVERFQDN
                              FQDN of satellite server - omit https://
  -f FROMCHANNEL, --from-channel=FROMCHANNEL
                              Source channel to clone errata from
  -d DESTCHANNEL, --dest-channel=DESTCHANNEL
                              Destination channel to clone errata to
  -e ERRATA, --errata=ERRATA
                              Comma separated list of errata to clone
```

Due to a known issue, the organization wishes to make RHSA-2013:0737 & RHBA-2013:0735 available to their systems. They'll use mergeErrata.py to clone that errata from the rhel-x86_64-server-6 channel to their Development channel (dev-rhel6-x86_64-server-6). This process would need to be repeated for the QA and Prod channels (qa-rhel6-x86_64-server-6) and (prod-rhel-x86_64-server-6) respectively.

```
./mergeErrata.py -l admin -p **** -s satellite.domain.com -f rhel-x86_64-server-6 -d dev-rhel6-x86_64-server-6 -e RHSA-2013:0737,RHBA-2013:0735
Attempting to merge the following errata from rhel-x86_64-server-6 to dev-rhel6-x86_64-server-6 ...
['RHSA-2013:0737', 'RHBA-2013:0735']
Errata successfully merged
```

Source:

```
#!/usr/bin/env python

"""
File: mergeErrata.py
Author: Rich Jerrido [rwj@redhat.com]
Version: 1.0.1
Last Modified: 03-Oct-2013
Purpose: Given two satellite channels (source & destination), merge a
(comma separated) list of errata.

Copyright 2013, Red Hat, Inc.

This program is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation; either version 2 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
GNU General Public License for more details.

"""

import getpass
import sys
import xmlrpclib
from optparse import OptionParser

parser = OptionParser()
parser.add_option("-l", "--login", dest="login",
    help="Login user for Red Hat Satellite/Hosted", metavar="LOGIN")
parser.add_option("-p", "--password", dest="password",
    help="Password for specified user. Will prompt if omitted",
    metavar="PASSWORD")
parser.add_option("-s", "--server", dest="serverfqdn",
    help="FQDN of satellite server - omit https://", metavar="SERVERFQDN")
parser.add_option("-f", "--from-channel", dest="fromchannel",
    help="Source channel to clone errata from", metavar="FROMCHANNEL")
parser.add_option("-d", "--dest-channel", dest="destchannel",
    help="Destination channel to clone errata to", metavar="DESTCHANNEL")
parser.add_option("-e", "--errata", dest="errata",
    help="Comma separated list of errata to clone", metavar="ERRATA")
(options, args) = parser.parse_args()

if not ( options.login and options.serverfqdn and options.fromchannel
    and options.destchannel and options.errata):
    print "Must specify login, server, fromchannel, destchannel and errata options.
See usage:"
    parser.print_help()
    print "\nExample usage: ./mergeErrata.py -l admin -p password -s
satellite.domain.com -f rhel-x86_64-server-6 -d 6 -e RHSA-2013:0737,RHBA-
2013:0735"
    sys.exit(1)
else:
    login = options.login
    password = options.password
    serverfqdn = options.serverfqdn
    fromchannel = options.fromchannel
    destchannel = options.destchannel
    errata = options.errata

if not password: password = getpass.getpass("%s's password:" % login)
```



```
SATELLITE_URL = "https://%s/rpc/api" % serverfqdn
SATELLITE_LOGIN = login
SATELLITE_PASSWORD = password

try:
    client = xmlrpclib.Server(SATELLITE_URL, verbose=0)
    key = client.auth.login(SATELLITE_LOGIN, SATELLITE_PASSWORD)
except xmlrpclib.ProtocolError as e:
    print "Unable to connect to %s . Error code: %s - %s " % (e.url,
        e.errcode, e.errmsg)
    sys.exit(1)
except Exception as e:
    print "Unable to login into the Satellite. Error: %s" % e.faultString
    sys.exit(1)

erratalist=errata.split(',')
try:
    print "Merging the following errata from %s to %s ..." % (fromchannel,
        destchannel)
    print erratalist
    servers = client.channel.software.mergeErrata(key, fromchannel,
        destchannel, erratalist)
    print "Errata successfully merged"
except Exception, e:
    print 'Unable to merge the requested errata. Error: ', e.faultString
client.auth.logout(key)

sys.exit(0)
```

Extended Update Support (EUS)

Extended Update Support adds a new dimension to Red Hat Enterprise Linux maintenance. EUS enables customers to keep using a specific minor release, and skip three subsequent minor releases, while remaining fully-covered by their Red Hat Enterprise Linux subscriptions.

The Red Hat Enterprise Linux Life-Cycle

The Red Hat Enterprise Linux life-cycle (<https://access.redhat.com/support/policy/updates/errata/>) consists of three phases, Production 1, 2,& 3:

Production 1

During the Production 1 phase, qualified Critical and Important Security errata advisories (RHSAs) and Urgent and Selected High Priority Bug Fix errata advisories (RHBAs) may be released as they become available. Other errata advisories may be delivered as appropriate. If available, new or improved hardware enablement and select enhanced software functionality may be provided at the discretion of Red Hat, generally in minor releases. Hardware enablement that does not require substantial software changes may be provided independent from minor releases at Red Hat's discretion. Minor releases will also include available and qualified errata advisories (RHSAs, RHBAs, and RHEAs). Minor releases are cumulative and include the contents of previously released updates. The focus for minor releases during this phase lies on resolving defects of medium or higher priority.

Production 2

During the Production 2 Phase, qualified Critical and Important Security errata advisories (RHSAs) and Urgent Priority Bug Fix errata advisories (RHBAs) may be released as they become available. Other errata advisories may be delivered as appropriate. If available, hardware enablement that does not require substantial software changes may be provided at the discretion of Red Hat, generally in minor releases. New software functionality is not available during this phase. Minor releases will also include all available and qualified errata. Minor releases are cumulative and thus include the contents of previously released minor releases and errata advisories, including those from Production 1 Phase. The focus for minor releases during this phase lies on resolving urgent- or high-priority bugs.

Production 3

During the Production 3 Phase, Critical Impact Security Advisories (RHSAs) and selected Urgent Priority Bug Fix Advisories (RHBAs) may be released as they become available. Other errata advisories may be delivered as appropriate. New functionality and new hardware enablement are not planned for availability in the Production 3 Phase. Minor releases with updated installation images may be made available in this Phase.

Red Hat Enterprise Linux Minor Releases and EUS

Minor Red Hat Enterprise Linux releases occur roughly every 6 months. All asynchronously-released errata are always incrementally based on the most recent minor release plus any errata subsequently released to that channel. It has therefore been necessary for customers to apply each and every minor release, including updates which only enable hardware, in order to stay up-to-date with the most recent security and bug fixes.

Extended Update Support relieves this burden for customers who are running large and/or mission-critical deployments of Red Hat Enterprise Linux, or who have long internal validation and testing cycles preventing them from keeping up-to-date with each minor release. The number of changes that the release of a new minor version brings with it can sometimes prove difficult and time-consuming for system administrators to manage. EUS offers extremely stable, long-term production environments with almost no "channel churn" (nearly zero unnecessary updates). With EUS, large-scale and mission-critical deployments are able to maintain a certain minor Red Hat Enterprise Linux release version while simultaneously receiving the benefits of critical-severity security advisories and urgent-priority bug fixes.

EUS is made available to customers for each minor release during the Production 1 Phase. EUS is available as a value added offering for some Premium subscriptions, and as an add-on for others. This allows it to be purchased either for all or a subset of systems, based upon need. Please contact your

Red Hat Sales Representative for more information on EUS. EUS provides an independent parallel life-cycle for each minor release of a period of 24 months as show below:

Typical EUS Life Cycle Stream – Red Hat Enterprise Linux 6

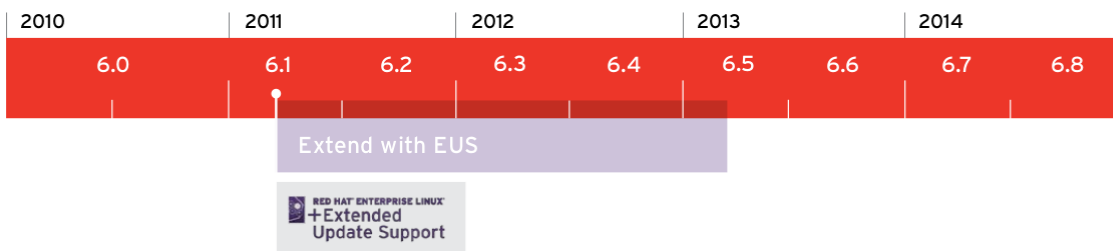


Figure B.1. Typical EUS Life Cycle Stream - Red Hat Enterprise Linux 6

Who should consider the Extended Update Support (EUS) add-on

Consider using the Extended Update Support add-on if:

- ▶ Your organization has applications that are only supported on a particular minor release
- ▶ Your organization has lengthy internal validation & certification processes that make it difficult to keep up with each minor release. This is common in many large organizations with shared IT organizations that serve multiple business units. EUS provides a means to put out a Standard Operating Environment (SOE) based upon RHEL 6.2.z, then 24 months later an SOE based upon 6.6.z. And lastly, an SOE based upon 6.10.z
- ▶ Your organization's tolerance for risk is low enough to warrant a much more conservative stream of errata.

Usage of EUS channels with this guide

While EUS provides a much more conservative stream of errata that is made available for systems, organizations still want the rigidity to be able to progress content from Development->QA->Production that is demonstrated via this guide. EUS channels can be used with this guide similarly to normal Base channels.

New Features of spacewalk-clone-by-date in Red Hat Satellite 5.6

Previously, spacewalk-clone-by-date used a different methodology when cloning errata than the Red Hat Satellite WebUI. The spacewalk-clone-by-date command used the "Issue Date" of an errata whereas the WebUI used the "Last Updated" date. The option **--use-update-date** has been added to the spacewalk-clone-by-date command. This option allows users to set the clone date to the date the cloned channel was last updated instead of the default issue date and aligns the behavior of the tool to be identical to that of the WebUI. This option can be added to the spacewalk-clone-by-date configuration files via the **use_update_date** directive as shown below:

```
{
  "username": "cadmin",
  "password": "*****",
  "assumeyes": true,
  "skip_depsolve": false,
  "security_only": false,
  "use_update_date": false,
  "blacklist": {
    "ALL": [
      "kernel*"
    ]
  },
  "removelist": {
    "ALL": [
      ""
    ]
  },
  "channels": {
    "rhel-x86_64-server-6": "dev-rhel6-x86_64-server-6",
    "rhn-tools-rhel-x86_64-server-6": "dev-rhn-tools-rhel-x86_64-server-6"
  }
}
```

Table C.1. use_update_date Options

Option	Meaning
true	Align the behavior of spacewalk-clone-by-date to be identical to the WebUI (e.g use the 'Update Date' of the errata)
false (Default)	Use the 'Issue Date' of the errata

Additional Reading & References

- ▶ Red Hat Satellite Product Documentation:
https://access.redhat.com/site/documentation/Red_Hat_Satellite/
- ▶ How am I supported on a specific RHEL release: <https://access.redhat.com/knowledge/articles/64664>
- ▶ Red Hat Lifecycle: <https://access.redhat.com/support/policy/updates/errata/>
- ▶ What strategies can be used to rollback packages:
<https://access.redhat.com/knowledge/solutions/65856>
- ▶ Payment Card Industry: Data Security Standard:
https://www.pcisecuritystandards.org/documents/pci_dss_v2.pdf
- ▶ ITIL: A guide to change management
http://www.ucisa.ac.uk/~media/Files/members/activities/ITIL/service/transition/change_management/ITIL_a_guide_to_change_management.pdf

Revision History

Revision 0-0	Fri Apr 12 2013	Rich Jerrido
Initial creation		
Revision 1	Fri Apr 19 2013	Rich Jerrido
Clarified usage of the 'to_date' JSON option. Added information regarding automating the 'satellite-sync' process Added license to mergeErrata.py Added additional information regarding Payment Card Industry example.		
Revision 2	Mon Apr 22 2013	Rich Jerrido
Added Appendix covering the EUS Lifecycle Add-on		
Revision 3	Mon Oct 3 2013	Rich Jerrido
Updated Product name from 'Red Hat Network Satellite' to 'Red Hat Satellite' Updated links to point to 'Red Hat Satellite' docs in the Customer Portal Added Appendix with Sample Patching Policy Added Appendix with New Features of spacewalk-clone-by-date Updated verbiage regarding EUS in line with the 2013 packaging change Updated EUS Chapter to be in line with the Customer Portal Fixed typographical & grammatical errors		