



SL/SLF Release Workflow

Assembled by Scientific Linux Distribution Support

This page intentionally left blank

Table of Contents

Executive Summary.....	5
1 Glossary of Technical Terms.....	5
Functional Workflow.....	7
1 Source Arrives.....	7
2 Build Source.....	7
3 Evaluate Source, Create Ticket, and Stage Appropriately.....	7
4 Evaluate Release.....	7
5 Resolve Tickets/Integrate and Test Internally.....	8
6 Change Management Request To Build.....	8
7 Open Change Task.....	8
8 Add Testers.....	9
9 Iterate Until Publication.....	9
10 Request To Go Live.....	9
11 Perform Publication.....	10
Workflow Diagrams.....	10
End User Testers.....	11
1 By Service.....	11
2 By Department.....	12

This page intentionally left blank

Executive Summary

This document attempts to detail the higher level activities of the Scientific Linux Release Workflow. It is based on the ITILv3 Release Management process as implemented at Fermi National Accelerator Laboratory.

Persons performing the actions outlined here are expected to be familiar with the SL/SLF Errata Build and Publishing Guide – CS-Doc-5454.

1 Glossary of Technical Terms

- A *Release* is a collection of items for publication as a unit. It may be a single item, such as an updated Kerberos configuration file, a minor release of Scientific Linux, such as SL 6.6, or a change to internal tools with external consequences. A *Release* is always under Change Management. For clarity a *Release* refers to this entry while a minor release update of a major release (RHEL 6.6), an update release (SCL 1.1), or a major release (RHEL 7) refers to an upstream collection which should be published as a single logical unit.
- The *Source* comes from our list of authorized providers. It may come in a number of forms.
- *Errata* refers to packages which provide: bug fixes, enhancements, and/or security patching outside of the upstream release cycle. One of these packages may come out at any time. The publication of *Errata* is considered WORK and not under change management at this time.
- A *Tree* is an installation source for Scientific Linux. It includes packages sufficient to install the operating system.
- The *Testing Repo* is available to all Scientific Linux systems. Its purpose is to provide easy access to the testing of specific packages independent of any operating system publication or minor release of Scientific Linux. Packages in the *Testing Repo* may not necessarily end up in any *Tree*.
- The *Private Tree* is our internal testing tree. It is not for public users or public testing.
- The *Rolling Tree* also known as the *Public Testing Tree* is our public location for testing new minor or minor releases of Scientific Linux. There is a *Rolling Tree* for each major version of Scientific Linux. At this time there is a '5rolling', '6rolling', and a '7rolling'.
- The *External Products Repo* is a location where each External Product build for Scientific Linux is published. These products may be independent of any particular version of Scientific Linux. These are under change management for non *Errata* publication.

- The *Production Tree* refers to the published *Tree* for a minor release. These are under change management for non *Errata* publication.

Functional Workflow

The following sections will attempt to abstract the particulars away from the *Release* process and focus on the higher level functions, their interactions, and procedural steps involved.

1 Source Arrives

Via an automated or manual process, an authorized *Source* provider publishes or provides an item of relevance to Scientific Linux.

2 Build Source

The provided *Source* should be built and staged for all relevant platforms.

3 Evaluate Source, Create Ticket, and Stage Appropriately

The result from the Build Source step should be evaluated:

- Is it a security update?
- Is it a non-security update?
- Is it part of an upstream minor release?

And the following question must be answered: How does this fit into any *Release* currently in progress?

Depending on the answers to the above questions a ENHANCEMENT ticket or DEFECT ticket should be opened and attached to a relevant *Release*.

Existing REQUEST or INCIDENT tickets can be transformed for attaching to this *Release*.

The assembled *Source* should be staged appropriately for publication.

4 Evaluate Release

The *Release* ticket should be examined to determine if it correctly encapsulates the update to be published.

- For upstream minor releases, this would encompass the elements from upstream as well as any local changes. Named such as 'SL 6.6'
- For update releases, it should cover the whole update. Named such as 'SCL 1.2'
- For single package fixes, it should include a verification that no other package fixes are required.
 - The *Release* name should either include the related DEFECT or ENHANCEMENT ticket or summarize the purpose of the update.

- Use your best judgment for ensuring clarity.

5 Resolve Tickets/Integrate and Test Internally

All associated ENHANCEMENT or DEFECT tickets should be completed

The changes should be integrated into the *Private Tree* or private product location. These should be tested following our internal procedures and tracked in TESTING tickets.

Any test failures should result in modification of related ENHANCEMENT or DEFECT tickets or the creation of new tickets to track the issues.

6 Change Management Request To Build

FOR THE SCIENTIFIC LINUX PRODUCTION TREE:

Once the *Release* has been successfully evaluated on the *Private Tree*, a Change Ticket should be created and placed into the “Request To Build” state. We are seeking to build the *Public Testing Tree* where we will publish the updates. It is expected that a simple sync from the *Private Tree* is the only required step.

So long as the projected changes only effect the *Public Testing Tree*, the “Request To Build” state is assumed by our line management. Go Live does not carry this assumption.

At this time we create these as MINOR changes. A projected Go Live date is selected, though it is not a firm commitment.

FOR THE EXTERNAL PRODUCTS REPOS:

Once the *Release* has been successfully evaluated internally, a Change Ticket should be created and placed into the “Request To Build” state. For the *External Products Repo* we place the product within the *Testing Repo*.

So long as the projected changes only effect the *Testing Repo*, the “Request To Build” state is assumed by our line management. Go Live does not carry this assumption.

For Upstream Minor Releases, we create these as MINOR changes. A projected Go Live date is selected, though it is not a firm commitment.

For fixes to previously released *Trees*, we create these as STANDARD changes. A projected Go Live date is selected, though it is not a firm commitment.

7 Open Change Task

We utilize Change Task tickets for following the workflow state of multi-step *Releases*.

FOR THE SCIENTIFIC LINUX PRODUCTION TREE:

Once the Change Ticket is “Approved To Build”, open a Change Task for publishing the change to the *Public Testing Tree*.

The Change Task Ticket is our method for tracking the publication date of Alpha, Beta, and Release Candidates.

FOR THE EXTERNAL PRODUCTS REPOS:

Once the Change Ticket is “Approved To Build”, open a Change Task for publishing the change to the *Testing Repo*.

The Change Task Ticket is our method for tracking the publication date.

8Add Testers

Our list of testers can be found in the section below.

Based on our agreement with our various stake holders, testing tickets are added. This is done via Service Now's templates system.

FOR THE SCIENTIFIC LINUX PRODUCTION TREE:

The testers are added to the specific Change Task covering the lifecycle (Alpha, Beta, etc).

FOR THE EXTERNAL PRODUCTS REPOS:

The testers are added to the Change Ticket itself as these products do not have a long public testing lifecycle.

9Iterate Until Publication

Steps “Resolve Tickets/Integrate and Test Internally”, “Open Change Task”, and “Add Testers” are repeated until we are ready for publication.

We iterate here for any Alpha, Beta, and Release Candidates being produced.

10Request To Go Live

Once all resolvable issues are corrected, a Go Live date is selected. The Change Ticket is then updated and placed into “Request To Go Live” state.

FOR THE SCIENTIFIC LINUX PRODUCTION TREE:

This happens along side of the last Release Candidate – typically RC2. The change task is still created for tracking the publication as testing may uncover errors which prevent the final release.

FOR THE EXTERNAL PRODUCTS REPOS:

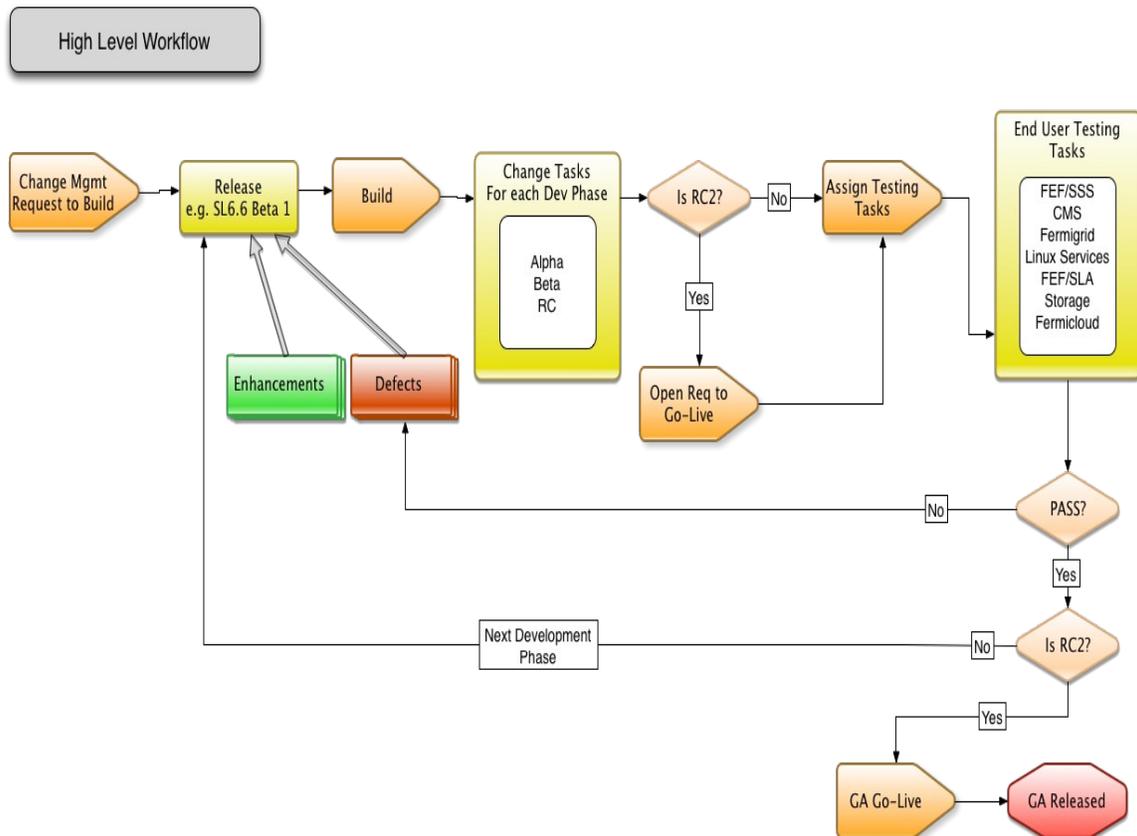
The request to Go Live represents our request to sync the private copy of the *External Products Repo* to the public copy.

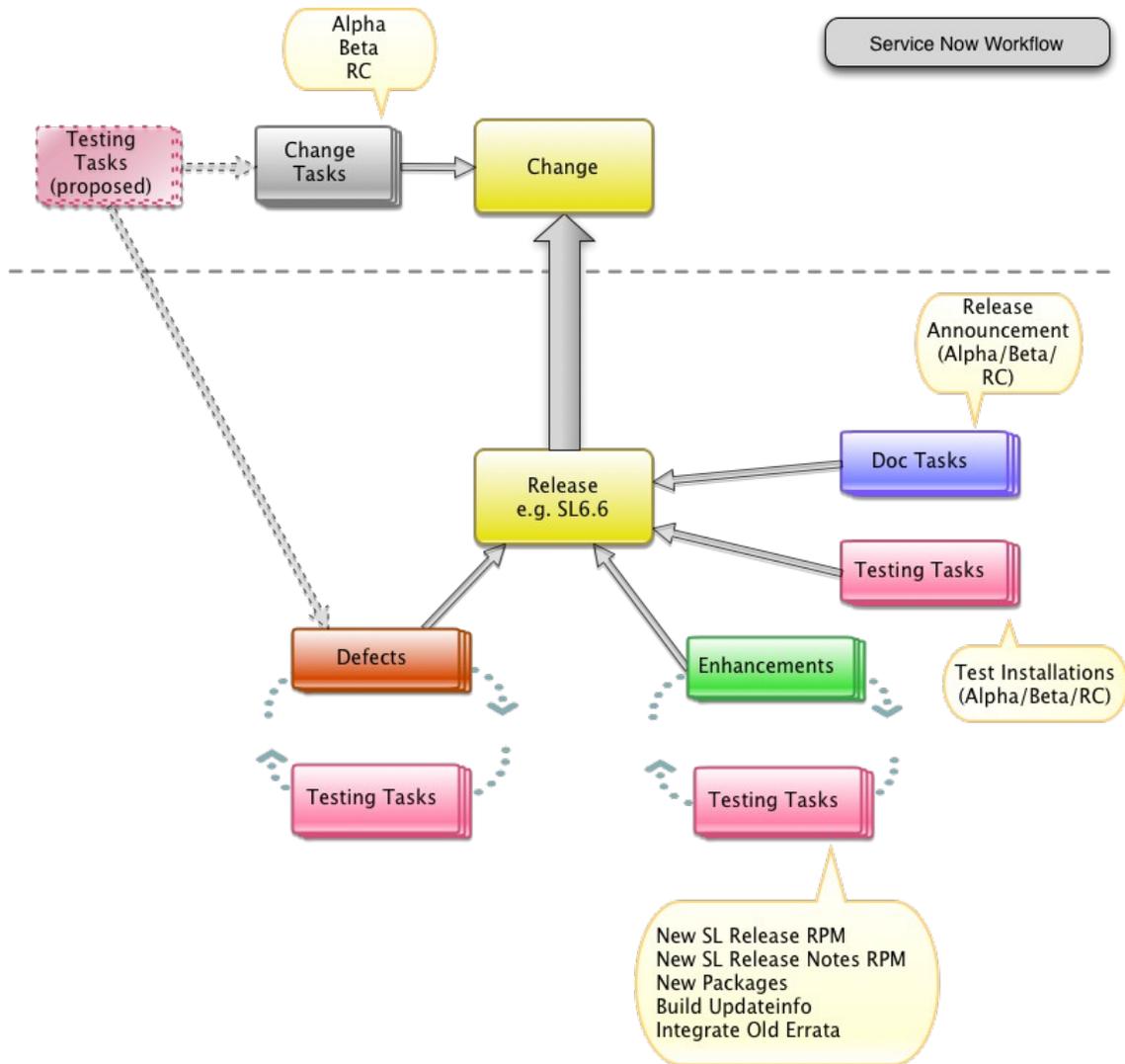
A Change Task is not needed for this sync as the Change Ticket itself describes the action.

11 Perform Publication

On the scheduled “Go Live Date”, the final production release is published.

Workflow Diagrams





End User Testers

There are two lists of Functional Testers here. They are not intended to overlap but they may.

1By Service

The following is a list of our primary stake holders by Service who will be invited to test releases:

- Authentication and Directory Services Services

- Service Owner: Al Lilianstrom
- Central Web Hosting Services
 - Service Owner: Peter Rzeminski II
- IT Server Hosting Services
 - Service Owner: Michael Rosier
- Virtual Server Hosting Services
 - Service Owner: Michael Rosier
- Scientific Collaboration Tools Services
 - Service Owner: Margherita Vittone Wiersma

2By Department

The following is a list of our primary stake holders by Department who will be invited to test releases:

- CCD/CSS
 - Head: Joe Klemencic
- SCD/ECF
 - Head: Glenn Cooper
- SCD/DCSO
 - Head: Lisa Giacchetti
- SCD/HPPC
 - Head: James N Simone
- SCD/SSA
 - Head: Kurt Biery
- SCD/LQCD
- SCD/FermiGrid