

Wind River VxWorks & Helix Virtualization Platform Defect Management Process

April 2020 V1

**Copyright Notice**

Copyright © 2020 Wind River Systems, Inc.

All rights reserved. No part of this publication may be reproduced or transmitted in any form or by any means without the prior written permission of Wind River Systems, Inc.

Wind River, Simics, Tornado, and VxWorks are registered trademarks of Wind River Systems, Inc. Helix, Pulsar, Rocket, Titanium Cloud, Titanium Control, Titanium Core, Titanium Edge, Titanium Edge SX, Titanium Server, and the Wind River logo are trademarks of Wind River Systems, Inc. Any third-party trademarks referenced are the property of their respective owners. For further information regarding Wind River trademarks, please see:

[www.windriver.com/company/terms/trademark.html](file:///C:\2019\Wind%20River\_Medical\Drafts\www.windriver.com\company\terms\trademark.html)

This product may include software licensed to Wind River by third parties. Relevant notices (if any) are provided for your product on the Wind River download and installation portal, Wind Share:

<http://windshare.windriver.com>

Wind River may refer to third-party documentation by listing publications or providing links to third-party websites for informational purposes. Wind River accepts no responsibility for the information provided in such third-party documentation.

**Corporate Headquarters**

Wind River  
500 Wind River Way  
Alameda, CA 94501-1153  
U.S.A.  
Toll free (U.S.A.): +1-800-545-WIND  
Telephone: +1-510-748-4100  
Facsimile: +1-510-749-2010

For additional contact information, see the Wind River website:

[www.windriver.com](file:///C:\2019\Wind%20River\_Medical\Drafts\www.windriver.com)

For information on how to contact Customer Support, see:

[www.windriver.com/support](file:///C:\2019\Wind%20River\_Medical\Drafts\www.windriver.com\support)

*Wind River Product Development Life Cycle*

April 2020

|  |  |  |
| --- | --- | --- |
| **Revision** | **Date** | **Description** |
| Ver. 1 | April 2020 | Initial release of document |

THIS DOCUMENT IS PROVIDED FOR INFORMATIONAL PURPOSES ONLY AND IS PROVIDED AS IS, WITH NO WARRANTIES, EXPRESS OR IMPLIED. WIND RIVER AND ITS LICENSORS DISCLAIM ALL WARRANTIES, EXPRESS, IMPLIED, AND STATUTORY INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE, AND NONINFRINGEMENT OF THIRD- PARTY RIGHTS. NO ORAL OR WRITTEN INFORMATION OR ADVICE GIVEN BY WIND RIVER OR ITS DEALERS, DISTRIBUTORS, AGENTS, OR EMPLOYEES SHALL IN ANY WAY INCREASE THE SCOPE OF THIS WARRANTY. WITHOUT LIMITING THE FOREGOING, WIND RIVER DOES NOT WARRANT THAT THIS DOCUMENT OR PERFORMANCE OF ANY WIND RIVER SOFTWARE WILL MEET CUSTOMERS’ REQUIREMENTS OR THAT CUSTOMERS WILL BE ABLE TO ACHIEVE ANY PARTICULAR RESULT FROM USE OR MODIFICATION OF THE THIS DOCUMENT OR ANY WIND RIVER SOFTWARE OR THAT THIS DOCUMENT OR WIND RIVER SOFTWARE IS FREE FROM, OR WILL OPERATE FREE FROM, ERROR. WIND RIVER PRODUCTS ARE COMMERCIAL OFF THE SHELF PRODUCTS, ARE NOT MEDICAL DEVICES, AND ARE NOT SPECIFICALLY DESIGNED FOR MEDICAL APPLICATION OR ANY OTHER PARTICULAR APPLICATION. WIND RIVER CUSTOMERS WHO ARE MEDICAL DEVICE MANUFACTURERS ARE SOLELY RESPONSIBLE FOR THE QUALITY AND SAFETY OF THEIR DEVICES, AND FOR SATISFYING ALL APPLICABLE REGULATORY REQUIREMENTS NECESSARY TO MARKET AND DISTRIBUTE SUCH MEDICAL DEVICES.

This Document is the property of Wind River Systems, Inc. This document, including the information contained herein, is copyrighted material, and includes Wind River Confidential Information. It is not to be used by or disclosed to any other individual or legal entity without the express written authorization of Wind River Systems.

**Contents**

[1 Introduction 6](#_Toc37782194)

[2 Process Overview 7](#_Toc37782195)

[2.1 Activities 7](#_Toc37782196)

[3 Submit Defect Procedure 9](#_Toc37782197)

[3.1 Steps 9](#_Toc37782198)

[3.2 Next Activity in Process 10](#_Toc37782199)

[4 Triage Defect Procedure 11](#_Toc37782200)

[4.1 Steps 12](#_Toc37782201)

[4.2 Variations 14](#_Toc37782202)

[4.2.1 Developer-conducted Triage 14](#_Toc37782203)

[4.2.2 Re-submission of On-Hold Defects 14](#_Toc37782204)

[4.3 Next Activity in Process 14](#_Toc37782205)

[5 Assign Defect Fix Procedure 15](#_Toc37782206)

[5.1 Steps 15](#_Toc37782207)

[5.2 Next Activity in Process 15](#_Toc37782208)

[6 Develop Defect Fix Procedure 16](#_Toc37782209)

[6.1 Steps 16](#_Toc37782210)

[6.2 Variations 17](#_Toc37782211)

[6.2.1 Fixing a defect in an external component 17](#_Toc37782212)

[6.2.2 Rerouting the defect to another Component 17](#_Toc37782213)

[6.3 Next Activity in Process 17](#_Toc37782214)

[7 Integrate Defect Fix Procedure 18](#_Toc37782215)

[7.1 Steps 18](#_Toc37782216)

[7.2 Variations 18](#_Toc37782217)

[7.3 Next Activity in Process 18](#_Toc37782218)

[8 Verify Defect Fix Procedure 19](#_Toc37782219)

[8.1 Steps 19](#_Toc37782220)

[8.2 Defect Verification Policy 19](#_Toc37782221)

[8.2.1 Policy 19](#_Toc37782222)

[9 Review Defect Procedure (Bug Board Review Procedure) 21](#_Toc37782223)

[9.1 Steps 21](#_Toc37782224)

[9.2 Definitions 22](#_Toc37782225)

[9.3 Bug Board Policy 22](#_Toc37782226)

[9.3.1 Policy 22](#_Toc37782227)

[10 Prepare Defect for Publication Procedure 24](#_Toc37782228)

[10.1 Steps 24](#_Toc37782229)

[10.2 Variations 25](#_Toc37782230)

[10.2.1 Modifying a Published Defect 25](#_Toc37782231)

[10.2.2 Removing a Published Defect 25](#_Toc37782232)

[10.3 Appendices 25](#_Toc37782233)

[10.3.1 Appendix A - Criteria for Publication 25](#_Toc37782234)

[10.3.2 Appendix B - Published Summary and Published Descriptions 26](#_Toc37782235)

[10.3.3 Appendix C - Style Guidelines 26](#_Toc37782236)

[10.4 Defect Publication Policy 27](#_Toc37782237)

[10.4.1 Policy 27](#_Toc37782238)

[Appendix A: Jira Defect Record Standard 29](#_Toc37782239)

[Overview Tab 29](#_Toc37782240)

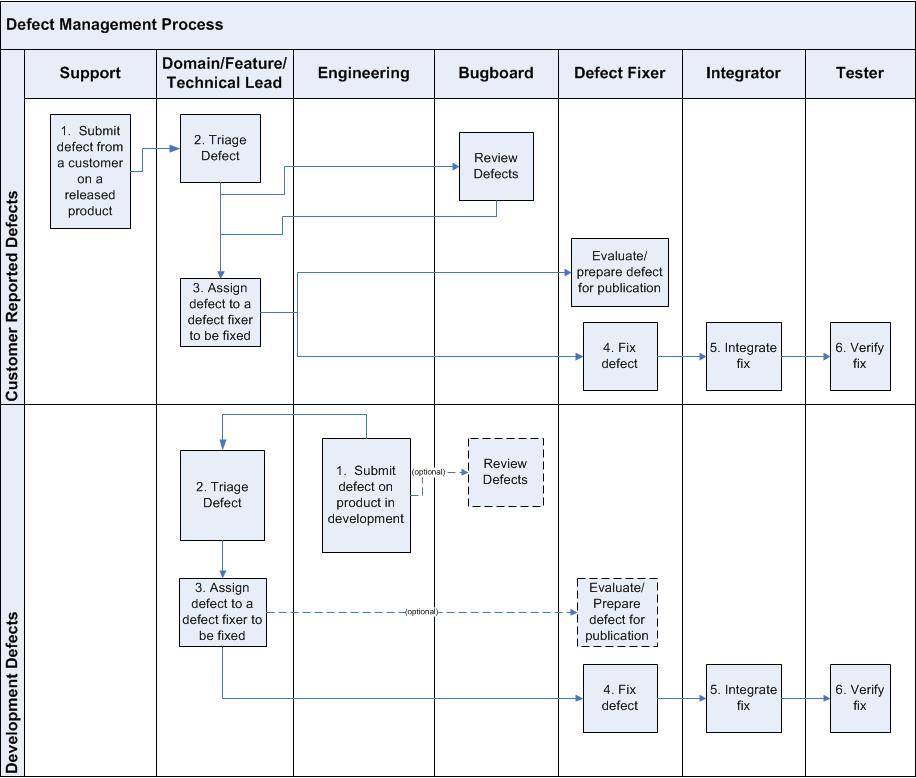
1. Introduction

This document describes the Wind River VxWorks and Helix Virtualization Profile (HVP) Defect Management Process. The process has similarities to the one described within the Wind River Linux Quality Management System (QMS) v1.8. The Engineering Program Manager (EPM) is responsible for ensuring that this process is followed.

1. Process Overview

|  |  |
| --- | --- |
| **Entry Criteria** | * Defect has been found |
| **Inputs** | * Defect Information |
| **Exit Criteria** | One of the following happens to the defect:   * The defect has been fixed and Status is set to “CLOSED” * The defect is resolved and Resolution is set to “Won’t Fix” * The defect has been determined not to be a defect and Resolution is set to “Rejected” or “Withdrawn” * The defect is determined to be a requirement, is reclassified as an “Enhancement Request” |
| **Outputs** | * Defect record has been updated * The defect fix |

* 1. Activities



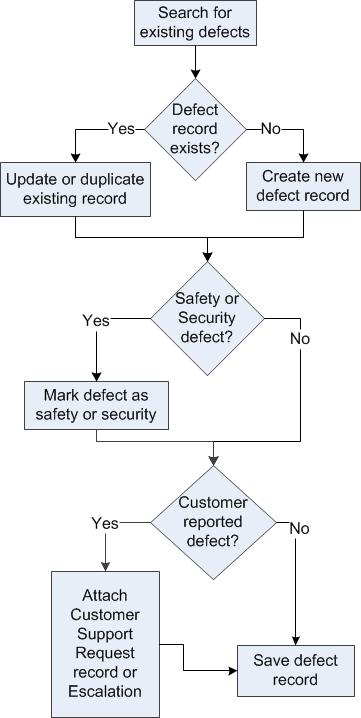
|  |  |  |
| --- | --- | --- |
| **Step #** | **Activity Name** | **Description** |
| 1 | Submit defect | Engineering or a Customer Support Engineer reports a product defect to the responsible engineering component team for resolution. |
| 2 | Triage defect | Using minimal effort, the Domain Lead (Feature/Technical Lead):   * Ensures new defect records have been properly routed and adequately documented * Weeds out non-defects by marking the Resolution as “Won’t Fix”, “Rejected” or reclassifying the Issue Type as “Enhancement Request” * Initiates timely action to address high priority defects, especially customer-reported defects.   **Note:** Anytime during the defect management process, a defect may be put “On Hold” status due to additional information required to move to the next state. |
| 3 | Assign a defect to a defect fixer to be fixed | The Engineering Manager or scrum team assigns defects to defect fixers (developer, tech writer, etc.). In the case of a customer reported defect, it may be necessary for the Domain Lead (Feature/Technical Lead) to assign the defect for resolution to ensure a timely response. |
| 4 | Fix defect | The assigned defect fixer (developer, tech writer, etc.) creates, tests, and checks-in a fix for the defect. |
| 5 | Integrate fix | The checked-in fix is integrated and integration tests are performed. Release build includes the integrated fix. |
| 6 | Verify fix | All fixes are verified on an “official” release build or other distribution image, according to the Defect Fix Verification Policy. |
| Parallel Activity to 3 | Review defects | The Bug Board reviews all open defects (both new and backlog) according to the Bug Board Policy. Once a defect has been resolved, it will not reviewed. |
| Parallel Activity to 2-6 | Evaluate/Prepare defect for publication | Preparing defects for publication is the responsibility of the defect fixer.  The decision to publish defects follows the Defect Publication Policy. |

1. Submit Defect Procedure

This procedure describes how to submit a defect. The filer of the defect is responsible for executing all the steps of this procedure.

|  |  |
| --- | --- |
| **Entry Criteria** | * Defect is discovered |
| **Inputs** | * Defect Information |
| **Exit Criteria** | * A new defect has been entered |
| **Outputs** | * Defect record in the “Open” Status has been submitted |

* 1. Steps



1. **When a defect has been found, before reporting it, search the defect database to see if a defect has been filed before for the same issue**.
2. **If an existing defect is found, update the record with any new information.**

**Note:** For customer support defect, filling in the *Found in Version* field is mandatory.

1. **If a defect record doesn’t exist, create a new defect record. The following fields are required when creating a new defect:**

* Project
* Issue Type (set to “Bug”)
* Summary
* Priority
* Severity
* Components
* Found in Versions
* Where Found

These additional fields should also be filled in, if known:

* Description
* Steps To Reproduce
* Workaround
* Host OS
* Processor Architecture
* Processor
* Environment

*Exception:* For confidential security information, a placeholder defect record may be created. This record will contain minimal information, until the issue is no longer confidential. Once the information is no longer confidential, the fields will be filled out as noted above.

**Note:** For definitions for specific fields in Jira, click on the ? icon next to the field within Jira.

**Note:** For Security related defects, fill out the fields in the Security Tab in Jira.

1. **If a defect was reported by a customer, and the submitter is in customer support, attach the Customer Support Request record. If the defect is part of an escalation, attach the Escalation record.**
2. **Save the defect record.**
   1. Next Activity in Process

|  |  |
| --- | --- |
| **Role** | **Activity** |
| Domain Lead (Feature/Technical Lead) | Triage Defect Procedure |

1. Triage Defect Procedure

The purpose of defect triage is to a) ensure new defect records have been properly routed and adequately documented, b) weed out non-defects, and c) initiate timely action to address high priority defects, especially customer-reported defects.

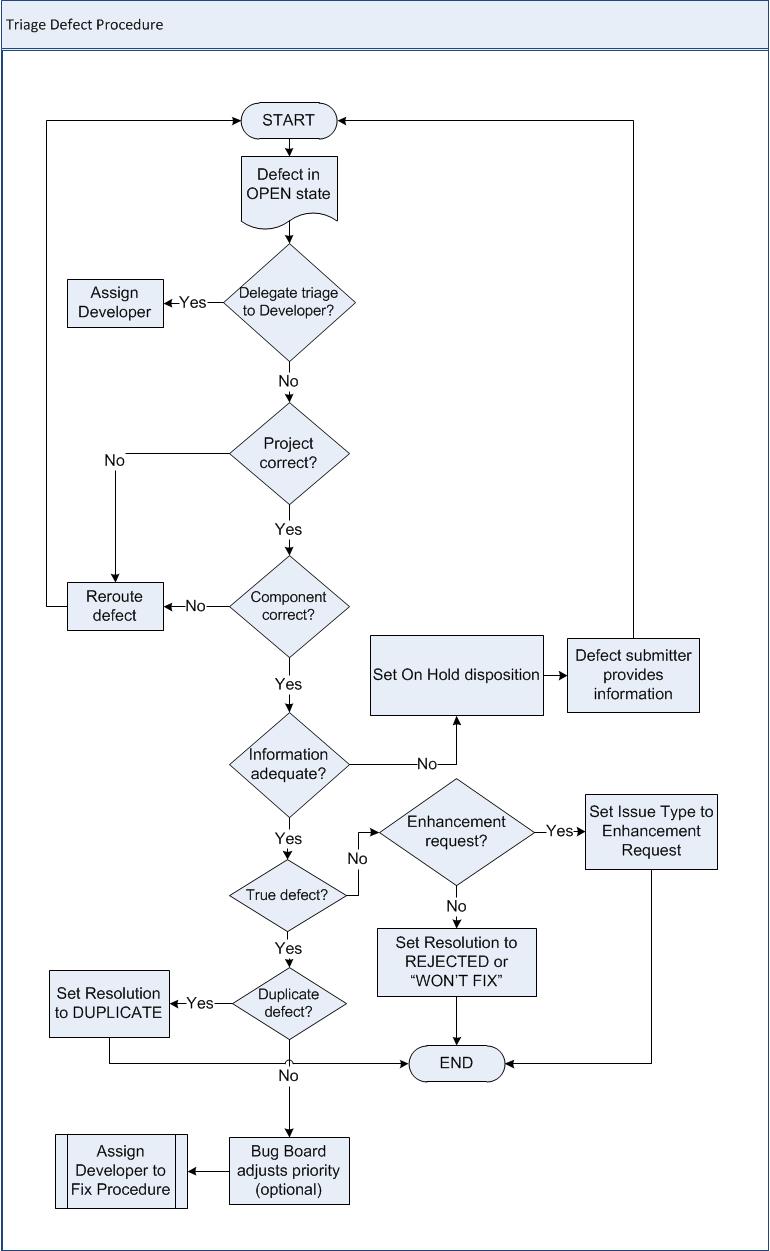
The focus of triage is to quickly assess the defect record and to defer a thorough technical investigation, including attempts at reproduction of the problem.

The defect record is taken at face value: that the submitter is competent in the use of the product and that the symptoms are accurately reported. This is especially essential in dealing with defects that are difficult to reproduce because we assume the defect exists even if we can’t reproduce it.

The Engineering Manager, Technical Lead, or Scrum team is responsible for this procedure. In the case of a customer reported defect, it may be necessary for the Domain Lead (Feature/Technical Lead) to assign the defect for a timely resolution.

|  |  |
| --- | --- |
| **Entry Criteria** | * A defect has been found and is in *Status* “Open” |
| **Inputs** | * Defect record in the Defect Tracking System (e.g., Jira) |
| **Exit Criteria** | * The Defect *Status* is in one of the following   + “Open” and assigned to defect fixer   + Set to “On Hold” with a reason provided   + “Resolved” with Resolution set to “Duplicate”, “Won’t Fix”, or “Rejected”   + Withdrawn * The defect may be re-categorized by setting *Issue Type* to “Enhancement Request” * The defect Priority and Severity may be adjusted |
| **Outputs** | * Updated defect record |

* 1. Steps



1. **Identify new defect record responding to either e-mail notifications, or running periodic queries, identify relevant defects in** *Status* **“Open”.**
2. **Examine the defect record** *Summary* **and** *Description* **to determine if they describe unexpected or incorrect behavior or properties. Ensure that the** *Description* **or** *Internal Description* **fields clearly establish the expected behavior, as criteria for a fix.**
3. **If the defect belongs to another Domain Lead, reroute it**
   * If the new component is in the same project
     + Click on *Edit*
     + Change the *Component* and *Assignee* fields
     + Click on *Update* to save changes
   * If the new component is in a different project
     + Click on *More Actions -> Move*
     + Select the *New* *Project*
     + Update any other fields as necessary
     + Click on *Move* to save the changes
4. **If the provided data is inadequate, return the defect to the submitter**
   * Take this course only if the provided data is inadequate to understand the symptoms or configuration.
   * To return for completion, click on *On Hold* button, set *On Hold Disposition* to “Waiting for reporter” and provide a reason
5. **If the reported defect is an Enhancement Request, re-categorize it**
   * Edit the record
   * Set *Issue Type* as “Enhancement Request”
   * Click on *Update* to save and exit from the record.
6. **If the reported defect is NOT a true defect, reject it**
   * This category includes non-defects.
   * **Usability issues that cause user error or confusion are true defects and should not be rejected.**
   * To reject the record, click *Resolve* and select *Resolution* as “Rejected” or “Won’t Fix”
     + Record the rationale in a *Comment* and click on *Resolve* button.
   * To withdraw the record, request the submitter to withdraw the defect
     + Record the rationale in a *Comment* and click on *Withdraw* button.
7. **If the defect is a duplicate of an existing defect, mark it as “Duplicate”**
   * To *Duplicate* the record, click *Resolve Issue* and set *Resolution* to “Duplicate”.
     + Click *More Actions -> Link*
     + Choose “duplicates” in the drop down for *This issue*
     + Enter the original defect issue number in the *Issue* field
     + Click *Link*
     + Click the *Close Issue* button - if that button is not available, click *Workflow > Close Issue*
     + Choose “Duplicate” as the *Resolution*
     + Click *Close Issue*
8. **Plan and take appropriate action. There are several options for completing this task:**
   * Do nothing now. Wait for the Bug Board to finalize the priority (see Bug Board Review Procedure).
   * Assign a defect fixer to a fix immediately (see Assign Defect Fix Procedure). This is appropriate for Customer-reported defects of high priority requiring immediate attention.
9. **A notification will be automatically sent to the submitter.**
   1. Variations
      1. Developer-conducted Triage

* Developers submitting defects against their own work products may perform their own triage, and assign a defect to themselves to be fixed.
  + 1. Re-submission of On-Hold Defects
* When defects are resubmitted, they recycle through triage. The focus of the second round is the same as the initial triage: to ensure the missing information has been provided to allow completion of the triage.
  1. Next Activity in Process

Depending on decisions described above:

|  |  |
| --- | --- |
| **Role** | **Activity** |
| Engineering Management/Domain Lead | Assign Defect Fix Procedure |
| Bug Board | Review Defect Procedure |

1. Assign Defect Fix Procedure

The Engineering Manager or scrum team assigns development of a defect fix to the defect fixer (Developer, Technical Publications Writer, etc.) for a specific release. In the case of a customer reported defect, it may be necessary for the Domain Lead (Component/Technical/Feature Lead) to assign a defect fix for a timely resolution.

|  |  |
| --- | --- |
| **Entry Criteria** | * Defects are in “Open” *Status* * The *Assignee* is the default assignee (e.g., Domain Lead) |
| **Inputs** | * Defect records in the defect tracking system |
| **Exit Criteria** | * The defect fixer has been assigned to do the fix |
| **Outputs** | * Defect assignment |

* 1. Steps

1. **Click on *Edit* to open the record for editing**
2. **Change the assignee in the defect to the defect fixer**
   1. Next Activity in Process

|  |  |
| --- | --- |
| **Role** | **Activity** |
| Defect Fixer | Develop Defect Fix Procedure |

1. Develop Defect Fix Procedure

When the defect fixer has acknowledged the defect is valid, the defect fix development process begins. The objective of the process is to ensure that the developed fix is correctly tracked in the defect tracking system and checked-in to Configuration Management system.

The defect fixer is responsible for this procedure.

|  |  |
| --- | --- |
| **Entry Criteria** | * Defects are in “Open” *Status* * Defect is assigned to defect fixer |
| **Inputs** | * Defect records in the defect tracking system |
| **Exit Criteria** | * Modifications checked in to the appropriate branch of the Configuration Management system * Defect fixer has tested the fix * Defect is in Status “Open”, either Priority 1 or 2, and may need to be released with the product, or to be reviewed by the Bug Board |
| **Outputs** | * Defect has been fixed with Status set to “Checked In”, and the details of the fix recorded * The defect fix |

* 1. Steps

1. **Understand the problem**

* Study the defect *Summary*, *Description*, *Internal Description*, and *Steps to Reproduce* to understand both the expected and observed behavior.
* Identify and resolve any ambiguities and gaps in the recorded descriptions, contacting the submitter if necessary, then *Edit* and *Update* the defect to record changes made.
* Reproduce the problem and isolate the cause.

1. **Reject a defect**

* If the reported defect is a true defect but will not be fixed, set it to Won’t Fix and enter a reason in the *Comment* field.
* To reject the defect, add justification for rejecting the defect and assign the defect back to the submitter. Request the submitter to withdraw the defect

1. **Optionally, consider alternative solutions, select the appropriate one, and estimate the effort.**

* If the estimate is unexpectedly large, notify the Component Lead (Domain/Technical/Feature Lead) before moving the defect record to the “In Progress” state.

1. **Develop the fix**

* Move *Status* to “In Progress” by clicking the “Start Progress” button
* Correct the defect
* Unit test the fix
* Update any related specifications or data affected by the fix

During development of the bug fix, the following change in status may be possible:

* A defect may be re-evaluated by the developer and Component Lead (Domain/Technical/Feature Lead) if they determine that the commitment to fix cannot be satisfied. In this case, the defect is returned to the Bug Board or defect originator for re-evaluation.
* If work on a defect has to be postponed, the defect is set to *On Hold* to notify Key Stakeholders of the temporary delay in developing the fix. (For example, developer temporarily reassigned to higher priority defect)

1. **Submit the defect fix according to the submission process**
2. **The** Release Build **process will automatically move fixed to the “***Checked in”* **state**

* Include any details regarding the fix.
  1. Variations
     1. Fixing a defect in an external component
* If the defect is to be resolved in an external component, such as Open Source or other component that is not directly under our control, the defect status is set to “On-hold” -> “Waiting for Vendor”
* The defect is then fixed in the external (non-product) location and made available to the team.
* When that code is integrated, there is no automatic updates to the defect records.
* When the defect filer/developer/tester identifies that the defect is no longer present, the defect record is manually moved to the “Fixed” resolution and a comment is added indicating why it was moved to fixed.
  + 1. Rerouting the defect to another Component
* If while working on the fix the defect fixer determines that the root of the defect is in another Component’s code, the defect fixer or Component Lead should reroute the defect to the correct team.
  + If the new component is in the same project, then the defect can be edited and the component and assignee changed.
  + If the new component is in a different project, the defect must be “Moved” to the appropriate project:
    - Select the defect to be moved
    - Click on *More Actions -> Move*
    - Select the new project and update any other fields as necessary
    - Click on *Move*
  1. Next Activity in Process

|  |  |
| --- | --- |
| **Role** | **Activity** |
| Integrator | Integrate Defect Fix Procedure |

1. Integrate Defect Fix Procedure

Once a checked-in fix is confirmed to be present on the integration branch, the defect record is updated. The procedure ensures that integration testing has passed before the defect can be marked “Resolved”/”Fixed”.

|  |  |
| --- | --- |
| **Entry Criteria** | * Code modifications have been completed and the developer has tested the fix * There is an integration branch to merge to * Modifications checked in to the appropriate branch of the version control system * The integration build has been performed * The integration tests for the fix have passed |
| **Inputs** | * The defect record is marked “Checked In” |
| **Exit Criteria** | * Integration test results are captured in the defect record * Code is merged |
| **Outputs** | * Defect record’s Status is set to “RESOLVED”/”FIXED” |

* 1. Steps

**The** Release Build **process will automatically move fixed records to “Resolved”/”Fixed”. This will set the “Fix Version” to the one that was built.**

* 1. Variations

In cases where the system cannot automatically move the fixed record to integrated, the Defect Fixer should monitor progress of their fix to ensure integration has completed successfully. They will manually move the record to “Resolved”/”Fixed” and assigned a Fix Version.

* 1. Next Activity in Process

|  |  |
| --- | --- |
| **Role** | **Activity** |
| Tester | Verify Defect Fix Procedure |

1. Verify Defect Fix Procedure

Verify that the fix has been properly integrated into either the development tree or the targeted release. If the product does not currently have release builds (i.e., still in development), it should be verified in the development tree. If there are release builds, it must be verified in the release build. Release builds incorporate prior fixes and their verifications, including those in the development tree, due to the iterative nature of the development process.

Verify that the fix has resolved the defect. Verification applies to all defects.

The individual who submitted the defect acts as tester and is responsible for this procedure.

|  |  |
| --- | --- |
| **Entry Criteria** | * Fix has been integrated * For release builds, there is a release build to test * Defect is ready for verification |
| **Inputs** | * Defect records are Integrated indicating that fixes are ready for verification * Updated defect record * For release builds, there is a release build to test |
| **Exit Criteria** | * The fix has been verified * The fix has failed verification and defect is reopened via the *Restart* button. |
| **Outputs** | * Defect record’s Status is set to “CLOSED” |

* 1. Steps

1. **Determine what fixes to verify**
2. **Obtain the development tree or current release build to test**
3. **Review the defect** *Description*, *Internal Description*, and *Steps to Reproduce*
4. **If there are duplicate defects, review the** *Description*, *Internal Description*, and *Steps to Reproduce* **of each duplicate.**
5. **Follow the** *Steps to Reproduce* **listed in the defect and any duplicate defects to test the fix.** Duplicate defects may list significantly different descriptions and ways to reproduce a defect.
6. **If verification fails**, reopen the record

* Click on *Restart* button
* Record any details about the test failure in the *Comment* field
* The defect fixer is notified of the failure.

1. **If verification passes**, close the defect record
   1. Defect Verification Policy
      1. Policy

The purpose of the fix verification policy is to ensure that important defects that are intended to be fixed in a release are fixed.

* All priority defects must be verified within 2.5 weeks of their resolution.

1. Review Defect Procedure (Bug Board Review Procedure)

The purpose of the Bug Board is to review all defects (both new and backlog), establish their Priority, provide a limited triage, and to respond to exceptions such as shifting priorities and revocation of commitments.

|  |  |
| --- | --- |
| **Entry Criteria** | * Unresolved defects * Defects require reevaluation in commitment or priority setting |
| **Inputs** | * Defect records in the Defect Tracking System (e.g., Jira) |
| **Exit Criteria** | * The defect *Status* is in one of the following:   + Set to Status “On\_Hold” and a reason provided   + “Resolved” with *Resolution* set to “Rejected”   + “Resolved” with *Resolution* set to “Duplicate”   + “Resolved” with *Resolution* set to “Won’t Fix” * The defect may be re-categorized by setting *Issue Type* to “Enhancement Request” * The defect *Priority* and *Severity* may be adjusted * Prior commitments for a fix may be reevaluated and adjusted * The defect may be labeled as “gating\_SRxxxx” and block the release indicated by SRxxxx * The defect may be labeled as “notgating\_SRxxxx” |
| **Outputs** | * When review completed, a comment or label may be applied in the defect record * Updated defect record |

* 1. Steps

1. An Open defect is acknowledged if the Bug Board perceives the defect as a bug or if the defect requires further investigation. If necessary, the Bug Board requests that an engineer investigate the defect, to decide if it really is a bug.

* A defect is rejected if the Bug Board (or delegate) determines the defect is not a bug. When a defect is rejected, a member of the Bug Board (or delegate) enters an explanation for the rejection in the Comment field.
* If the Bug Board, or investigating engineer determines the defect to be an enhancement request, the Issue Type will be reset to “Enhancement Request”.
* If the Bug Board does not have enough information about the defect in order to proceed, the defect’s Status is set to “On Hold” -> “Waiting for Reporter”
  + If a defect is marked “On Hold” -> “Waiting for Reporter”, a member of the Bug Board enters a request for the missing information into the comment field. The defect originator will be notified by e-mail that a change has occurred in the defect, and will be responsible for completing the missing information. Once the missing information is supplied, the defect originator reopens the defect, which causes the defect to cycle through the triage procedure once again.
* If the Bug Board deems that the defect is to be postponed, the defect’s Status is set to “On Hold” -> “Backlog”
* If the defect is deemed to be a duplicate, the defect is resolved as “Duplicate”, and is linked back to the original defect.

1. The Bug Board reviews (or assigns if missing) the Priority for each new defect.

* Review the *Description* and *Steps to Reproduce* descriptions, as well as initial *Priority* classifications to determine an appropriate *Priority* setting.
* **For Customer Defects, do not change the** *Priority* **setting without first consulting with the Customer Support submitter, since this may delay appropriate relief and resolution.**

1. The Bug Board reviews the gating status of defects

* If the Bug Board deems that the defect prevents acceptance of the Release Criteria, the label “gating\_SRxxx” is added to the defect.
  + Gating defects are reported at the program level and closely monitored as development approaches a release.
* If stakeholder signoff has been provided, the Bug Board may replace the label “gating\_SRxxx” with “notgating\_SRxxx” to indicate the defect is no longer gating a release.
  + A member of the Bug Board enters the rationale for removal of the gating label into the comment field.
  1. Definitions

**What is a gating (defect)?**

A gating defect is one that meets any of the following criteria:

* Blocks a test suite from being run
* Blocks development (there is a dependency by development on having a fix implemented)
* Has a P1 or P2 priority
* Causes a Nightly Regression Test failure
* Advocated by a member of the Bug Board (E.g., PLM, CSO, customer representative)

**Who can remove gating keywords from a defect?**

A member of the Bug Board at the time of review and with rationale provided into the comment field.

**Who is a member of the Bug Board?**

Bug Board membership is fluid throughout a release. Reviews may take place informally via email or formally in meetings. Stakeholder participation increases as the development cycle approaches release. Typical stakeholders include developers, testers, customer support engineers, product line managers, field engineers, architects, and engineering program managers.

* 1. Bug Board Policy
     1. Policy

The purpose of the Bug Board is to review all unresolved Defects (both new and backlog) that may affect customers and establish whether they should gate the release. The Bug Board also responds to exceptions such as shifting priorities and revocation of commitments.

The Bug Board must review every unresolved customer defect affecting the target release at least once and prioritize it. The Bug Board may also review other product affecting defects at their discretion.

The Bug Board is the final authority on whether a defect is gating. The Defect gating label endorsed by the Bug Board may not be modified by a lower authority. For the purpose of endorsing the gating label, a quorum is required, consisting of empowered representatives from Engineering, and one other group (e.g., Product Marketing, Field and Customer Support).

1. Prepare Defect for Publication Procedure

Decide if defect is suitable to be published; record the decision, and if it is to be published, prepare and review the published information. Defects are published by the Defect Fixer.

Which defects must be published is defined in the Defect Publication Policy.

|  |  |
| --- | --- |
| **Entry Criteria** | * A defect is in a product released to customers. The defect may be reported by a customer, or found during development and not fixed before the product was released. |
| **Inputs** | * Defect record has *Publish to OLS* field set to “To be determined” or “Ready for Review” |
| **Exit Criteria** | * Customer Support and Development reported defects are marked as “Reviewed - Publish” and has Publication Hierarchy set * Defects not affecting customers are marked as “Never shipped to customer” |
| **Outputs** | * Defect record is updated with *Publish to OLS* field updated * Defect is updated on WR Support Network (WRSN) |

* 1. Steps

1. **Review defects with the** *Publish To OLS* **field set to “None, “To be determined” or “Ready for Review”**
2. **Change the** *Publish To OLS* **field to “Reviewed - Publish”, “Reviewed - Do NOT publish” or “Never shipped to customer”. Refer to the criteria for publication in the** Defect Publication Policy**.**
3. **For defects to be published:**

* Set the information to be published: Published Summary, Published Description, and Publication Hierarchy. See Appendices B and C.
* Review the information to be published: *Priority*, *Component/s*, *Fix Version*, *Steps to Reproduce*, *Workaround*, *Host OS*, and *Processor Architecture*
* Update the published fields as necessary
* Set *Publish to OLS* as “Reviewed - Publish”
* Click on *Update* to save the record

1. **For defects not to be published:**

* Set the *Publish to OLS* field to “Reviewed - DO NOT Publish”. Add a comment to explain why the defect report will not be published.

1. **For defects that do not affect customers:**

* Defects which do not affect customers (not in a released version or fixed prior to release) should be marked as “Never shipped to customer”

1. **When the Criteria for Publication in Appendix A are met, the defect record will be automatically published.** How a defect can become unpublished again is described in “Removing a Published Defect” in the Variations section below.
   1. Variations
      1. Modifying a Published Defect

Any changes made to a published defect will automatically update to WR Support Network (WRSN) without going through the review process again. Modification to OLS published fields should be checked for accuracy before saving any changes.

The following fields are published to WR Support Network (WRSN):

* *Published Summary*
* *Published Description*
* *Severity*
* *Component/s*
* *Fix Version*
* *Steps to Reproduce*
* *Work Around*
* *Host OS*
* *Processor Architecture*

If the changes to the published fields are significant, the defect should be submitted for review by setting the *Publish to OLS* field to “Ready for Review”, which will cause the defect to be removed from WR Support Network (WRSN) until it has been reviewed.

* + 1. Removing a Published Defect

A published defect is removed from WR Support Network (WRSN) if any one of the following is true:

* The *Publish To OLS* field is NOT set to “Reviewed - Publish”
* The *Issue Type* is reclassified as “Enhancement Request”
* The *Publication Hierarchy* does not have at least 2 levels.
  1. Appendices
     1. Appendix A - Criteria for Publication

All of the following conditions must be met for a defect to be published on WR Support Network (WRSN). When these criteria are met, the defect is automatically published.

* The Issue Type = Bug
* Publish To OLS = Reviewed - Publish
* The Publication Hierarchy has at least 2 levels
* Status and resolution matches one of the following combinations:

|  |  |
| --- | --- |
| Status | Resolution |
| On Hold and On Hold Disposition = Backlog | None |
| In Progress | None |
| Checked In | None |
| Resolved | Fixed or Won’t Fix |
| Closed | Fixed or Won’t Fix |

See [Jira FAQ](https://jive.windriver.com/docs/DOC-22124#jive_content_id_What_records_are_published_to_Knowledge_Library_from_Jira) for more information.

* + 1. Appendix B - Published Summary and Published Descriptions

The **Published Summary** becomes the **Title** of the defect when it appears on WR Support Network (WRSN).

* The goal is to provide a clear description of the symptoms resulting from the defect so that customers searching through published defects can quickly determine whether they are seeing this defect in their system or project

The **Published Description** becomes the **Description** of the problem on WR Support Network (WRSN).

* Provide a clear description of problem with additional detail around symptoms and how the problem was fixed
* If applicable, an indication that a workaround is available.
* If this is the case, the workaround must be documented in the Workaround field.
* If the defect is not to be fixed then the reason why this determination has been made must be documented.
* Patch availability or information indicating the release version the defect was fixed in (not internal code names, but actual release versions)

The **Workaround** remains the **Workaround** field on WR Support Network (WRSN)

* If a Workaround is found for the problem exists, then a Workaround description should be created and published.
* Any published Workaround should be broadly applicable and must be tested by engineering.
  + 1. Appendix C - Style Guidelines

Published fields must be reviewed and sanitized. In particular:

* All references to specific customers must be removed
* Any embarrassing, offensive or derogatory language must be removed
* Avoid forward-looking information/data. Avoid making promises of Wind River action. Remember that a public transaction will live for many years.
* Convey sufficient and useful information. The key here is to avoid the call into the Call Center. The Published Description should be complete enough for the customer to determine if the issue affects them. By all means, avoid dangling a juicy tidbit that prompts frantic calls into the Call Center. It is perfectly acceptable to put test cases in a public transaction if that helps convey the information.
* Product names should be spelled and formatted correctly.
* All product names are capitalized.
* Some product names have internal capitalization (such as ScopeTools).
* Spell out product names (do not use WB for Workbench, PS for ProfileScope, and so on).
* When discussing Operating Systems:
  + "VxWorks" should be used when discussing the product.
  + "vxWorks" should be used only when discussing a specific image file.
  + "vxworks" should never be used
  + Linux, Windows, and other operating system or third-party product names should be capitalized appropriately.
* All fields should contain complete sentences whenever possible. They should begin with capital letters and end with periods.
* The *Steps To Reproduce* field should contain numbered steps. Each step should be a complete sentence whenever possible.
* The published bug description must NEVER include project names, CCM/PS references, references to older WR products and/or discussions explaining that we made a mistake in a port.
* The description should be re-written to explain the component that is broken and the symptoms of the failure. All other details should be removed unless relevant to the external reader.
* The Subject from cloned defects includes the word ‘cloned’, which should be removed.
* Customers will not see the *Internal Description* or *Comment* fields. Use these fields for any mention of attached logs or screenshots, host/target configuration information, specific tests run, or any other internal Wind River information.
  1. Defect Publication Policy
     1. Policy

1. Defects in released products are to be reviewed for publication if they meet any of the following criteria:

* Open defects with a label of “gating\_SRxxx”

**Note:** There should be zero defects label as “gating\_SRxxxx” upon product release.

* Open P2 defects labelled “notgating\_SRxxxx”

**Note:** All P2 defects with a label “notgating\_SRxxxx” must have been reviewed by the Bug Board prior to product release.

* All customer-report defects, except those with a *Resolution* of “Rejected” or “Withdrawn”

1. Some defects may be exempt from publication if the defect could reasonably be expected to expose a customer, partner, or Wind River to unnecessary risk. Included under this exception are defects for which publication would:
   * Expose a problem related to a non-public security alert or other security advisory agency, or
   * Compromise pending litigation, or risk legal liability
   * The quality of the information, or the nature of the defect itself, makes it unsuitable for publication, as determined by the Bug Board.
2. The rationale for not publishing a defect must be recorded in each instance.
3. The following defect record text fields are published.

* *Published Summary*
* *Published Description*
* *Severity*
* *Component/s*
* *Fix Version*
* *Steps to Reproduce*
* *Workaround*
* *Host OS*
* *Processor Architecture*

Appendix A: Jira Defect Record Standard

The Jira defect tracking tool is located at: <http://jira.wrs.com>.

The following table is a list of the fields in a Jira defect record and the description of the fields.

|  |  |  |
| --- | --- | --- |
| **Field** | Description |  |
| **Project** | Jira Project. |  |
| **Issue Type** | **Type** | **Description** |
|  | **Bug** | The terms Bug and Defect are used interchangeably. A defect is defined as a failure of a work product to meet either:   * Requirement * Design * Standard * Peer review checklist item * Document template * Clarity |
|  | **Enhancement Request** | An enhancement request is additional functionality beyond the product requirements. |

Overview Tab

|  |  |  |
| --- | --- | --- |
| **Summary** | A brief one-line summary of the issue. For example, “Red Angry Nerd is scary.” | |
| **Priority** | **Priority** | **Definition** |
|  | P1 | Highest priority. Indicates that this issue takes precedence over all others. |
|  | P2 | Indicates that this issue is causing a problem and requires urgent attention. |
|  | P3 | Indicates that this issue has a significant impact. |
|  | P4 | Indicates that this issue has a relatively minor impact. |
| **Severity** | **Severity** | **Definition** |
|  | Critical | A **Critical** issue involves one or more of the following conditions:   * data loss or corruption * Loss of operations * Failure of customer devices that are in commercial operation with end customers * ‘work stop’ situation for customer or end customer * Grave impact to business operations * Incorrect code is generated with no warning or error message |
|  | Severe | A **Severe** issue involves one or more of the following conditions:   * Loss of major functionality in product * Work is severely impacted but can continue * Incorrect code is generated but a warning or error message is generated * Product or customer device crashes but product or device may be restarted * Business operations continue but are significantly impacted * Loss of data or data corruption that is reversible |
|  | Standard | A **Standard** issue involves one or more of the following conditions:   * Problem encountered, but systems are operational and development continues * Product or customer device issues an error message but product or device may be restarted * Generated code is correct but not ideal (i.e. optimization not done) * Business operations continue with minimal impact * Significant error in documentation, or feature documentation missing * May have reasonable workaround available |
|  | Trivial | A **Trivial** issue involves one or more of the following conditions:   * Minor documentation error, or feature documentation needs updating * No significant loss of functionality in product * Business operations are minimally affected * No data loss or corruption * Cosmetic or nuisance issues |
| **Found in Version** | Project version for which the issue is (or was) manifesting. | |
| **Fix Version** | Project version in which the issue was (or will be) fixed. | |
| **Description** | A detailed description of the issue. (published on OLS) | |
| **Internal Description** | A description not published on On Line Support. | |
| **Workaround** | How to get around the bug. (published on OLS) | |
| **Component(s)** | Project component(s) to which this issue relates. | |
| **Assignee** | The person to whom the issue is currently assigned. | |
| **Publish to OLS** | * None * To be determined * Reviewed – Publish * Reviewed – Do NOT Publish * Never Shipped to Customer | |
| **Label(s)** (*if applicable*) | Labels to which this issue relates. Bug Board approval, marking customer specified defects and other activities are handled with labels. | |
| **On Hold Disposition** | **The reason an issue has a status of “On Hold”. Values:**   * Incomplete * Waiting for Customer * Waiting for Vendor * Waiting for Submitter * Submitted to Open Source | |