Bug/Defect Tracking

- Bug Reporting Template
- Task Filing Template
- Defect Tracking and Reporting Flow Chart
- Bug Reporting Template

Bug Reporting Template

Who can report Defect/Bug?

Anyone came across the bug can report bug/defect and put that on the Project/Product Backlog.

After reporting the Bug QA team can organize the Triage meeting to set the priority and severity of that bug. Development Team lead, QA Team lead and Project Manager decide in which sprint that bug will be fix or that bug needs to be hotfix.

Note: If any team member other than QA found the bug/defect they have to assign that bug/defect to the QA of their project to verify whether that is bug/defect or not.

Template

Bug Title

A bug title is read more frequently than any other part of the bug report and should be clear and concise.

Everybody should be able to determine what the bug is without having to open the entire report.

Description

A well-defined bug description allows for an easy understanding of the bug and what happens.

Platform/Environment/Version/Context

It is also essential to include what operating system, browser configurations, and versions are being utilized when the bug is present.

Roles

Mention the in which user the bug has been replicated. (User roles)

Step to Reproduce

This provides clear instructions on how the bug can be reproduced by your developers.

Expected Results

Mention what is the expected results

Actual Results

Mention what is actually happening currently in that bug

Screenshot / Attachments / Screen Recording

Include a clear screenshot at the point of failure and when the bug occurred.

Example

Currently Playing shows data after flashing the ISO

Details

Context

All

Roles

- Have Permission to Flash Device
- Network Manager

Step to Reproduce

- Flash the Device which is already listed in the Salt Accepted list and working fine. which has Campaign playing
- Connect to the Internet
- Device now listed in the Denied Group
- Accept the Device in Denied Group
- Navigate to the Network Page
- Verify the Data on the Currently Playing & Last Sent

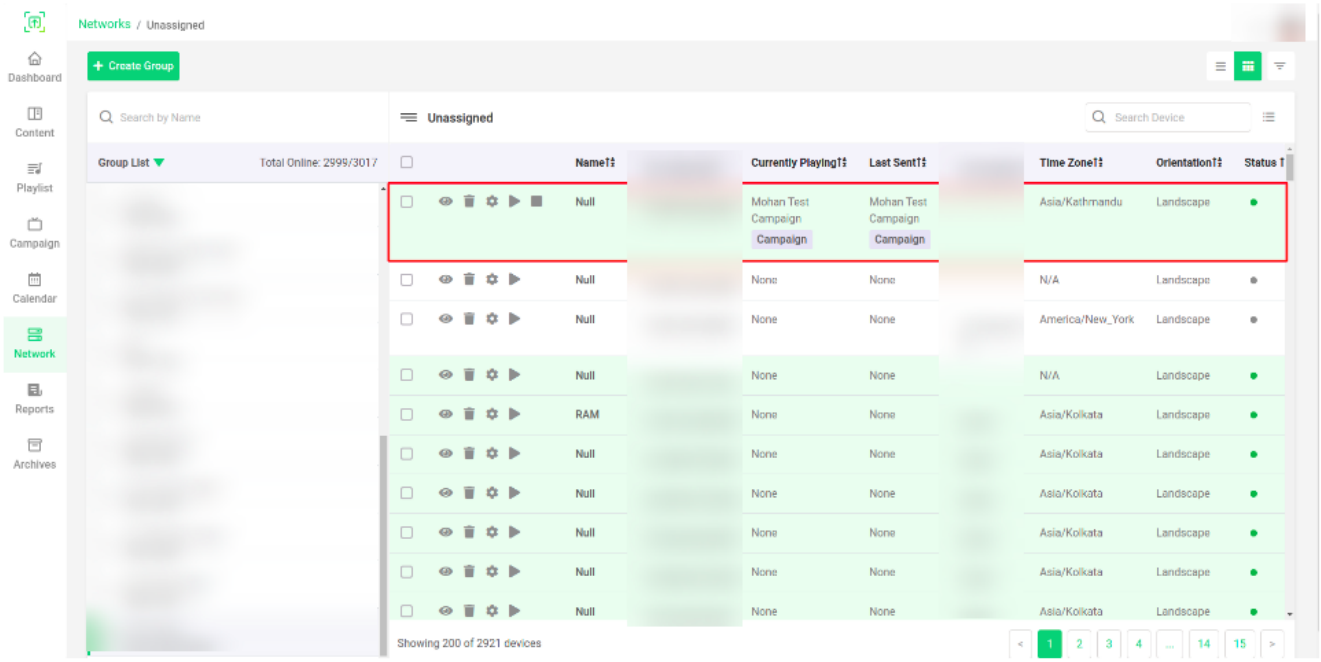
Expected Results

The currently playing, last sent data must be shown correctly in the UI

Actual Results

Previous data still remains on the currently playing after flashing the device

Attachments



Edited 4 days ago by Kishan Bhatta

Fig: Bug Example

Task Filing Template

Task Title

Task Title must be clear and concise. Any user can understand the task by just reading the task title.

Description

A well-defined task description allows for an easy understanding of the task.

Acceptance Criteria (AC)

Acceptance Criteria are the conditions that a software product must meet to be accepted by a user, a customer, or the system.

AC can be written in different formats. There are two most common ones, and the third option is to devise your own format:

- **Scenario-oriented (Given/When/And/Then):**

The scenario-oriented format is the acceptance criteria type that comes in the scenario form and illustrates each criterion.This approach was inherited from behavior-driven development(BDD) and provides a consistent structure that helps testers define when to begin and end testing a particular feature. It also reduces the time spent on writing test cases as the behavior of the system is described upfront.

It is approached through the *Given/When/Then* (GWT) sequence that looks like this:

1. *Given* some precondition
2. *When* I do some action
3. *Then* I expect some result

The acceptance criteria template includes the following statements:

1. Scenario – the name for the behavior that will be described
2. Given – the beginning state of the scenario
3. When – specific action that the user makes
4. Then – the outcome of the action in “When”

5. And – used to continue any of three previous statements

Example:

User Story: As a user, I want to be able to recover the password to my account, so that I will be able to access my account in case I forgot the password.

Scenario: Forgot password

- **Given:** The user navigates to the login page
- **When**: The user selects *<forgot password>* option
- **And**: Enters a valid email to receive a link for password recovery
- **Then**: The system sends the link to the entered email
- **Given:** The user receives the link via the email
- **When:** The user navigates through the link received in the email
- **Then:** The system enables the user to set a new password

- **Rule-oriented (Checklist)**

The rule-oriented form entails that there is a set of rules that describe the behavior of a system. Based on these rules, we can draw specific scenarios.Usually, criteria composed using this form look like a simple bullet list.

Example:

User story: As a user, I want to use a search field to type a city, name, or street, so that I could find matching hotel options.

Basic search interface acceptance criteria

- The search field is placed on the top bar
- Search starts once the user clicks “Search”
- The field contains a placeholder with a grey-colored text: “Where are you going?”
- The placeholder disappears once the user starts typing
- Search is performed if a user types in a city, hotel name, street, or all combined
- Search is in English, French, German, and Ukrainian
- The user can’t type more than 200 symbols
- The search doesn’t support special symbols (characters). If the user has typed a special symbol, show the warning message: “Search input cannot contain special symbols.”

- **Custom formats**

We can invent our own acceptance criteria given they serve their purposes, are written clearly in plain English, and can't be misinterpreted.

Example: Strong Password Acceptance Criteria

Data	Expected Result	Expected Message
Aa$5777	Fail	Too Short
AAbbDD11	Fail	No Specific Characters
$$$aaa111	Fail	No Upper Case
GGG$$$333	Fail	No Lower Case
SSS***hhhhhh	Fail	No Numbers
CharcatersMoreThan11$$$	Fail	Max Password Length is 255
This@ISGood11	Pass	
P@ssword11%	Pass	

Definition of Done (DoD)

The definition of done (DoD) is when all conditions, or acceptance criteria, that a software product must satisfy are met and ready to be accepted by a user, customer, team, or consuming system. It lowers rework, by preventing user stories that don't meet the definition from being promoted to higher level environments. It will prevent features that don't meet the definition from being delivered to the customer or user.

User Stories

The most common use of DoD is on the delivery team level. Done on this level means the Product Owner reviewed and accepted the user story. Once accepted, the "done" user story will contribute to the team velocity.

User Story DoD Examples:

- Unit tests passed
- Code reviewed
- Acceptance criteria met
- Functional tests passed
- Non-Functional requirements met
- Product Owner accepts the User Story

Features

Done on this level means it qualifies to add to a release. Not all user stories need to be completed. Rather, it means the feature may be sufficient to satisfy the need. Once accepted, the done feature will contribute to the release velocity.

Feature DoD Examples:

- Acceptance criteria met
- Integrated into a clean build
- Promoted to higher level environment
- Automated regression tests pass
- Feature level functional tests passed
- Non-Functional requirements met
- Meets compliance requirements
- Functionality documented in necessary user user documentation

Epics

Done on this level refer to a organizational strategic priority, portfolio plan item, or some other collection of features that satisfied a market need. Not all user stories or features need to be completed. Rather, the epic is sufficient to satisfy the need. Once accepted, the done epic will contribute to throughput calculations to see if the supply is in balance with demand.

Epic DoD Examples:

- Non-Functional requirements met
- End-to-end integration completed
- Regression tests pass
- Promoted to production environment
- Meets defined market expectations

Design Mockup/wireframe as a Attachments

If needed the design mockup/wireframe must be attached for a better understanding of the task and the UI flow.Expected Message

Defect Tracking and Reporting Flow Chart

Defect Tracking and Reporting Flow chart

The following flowchart depicts Defect Tracking Process:

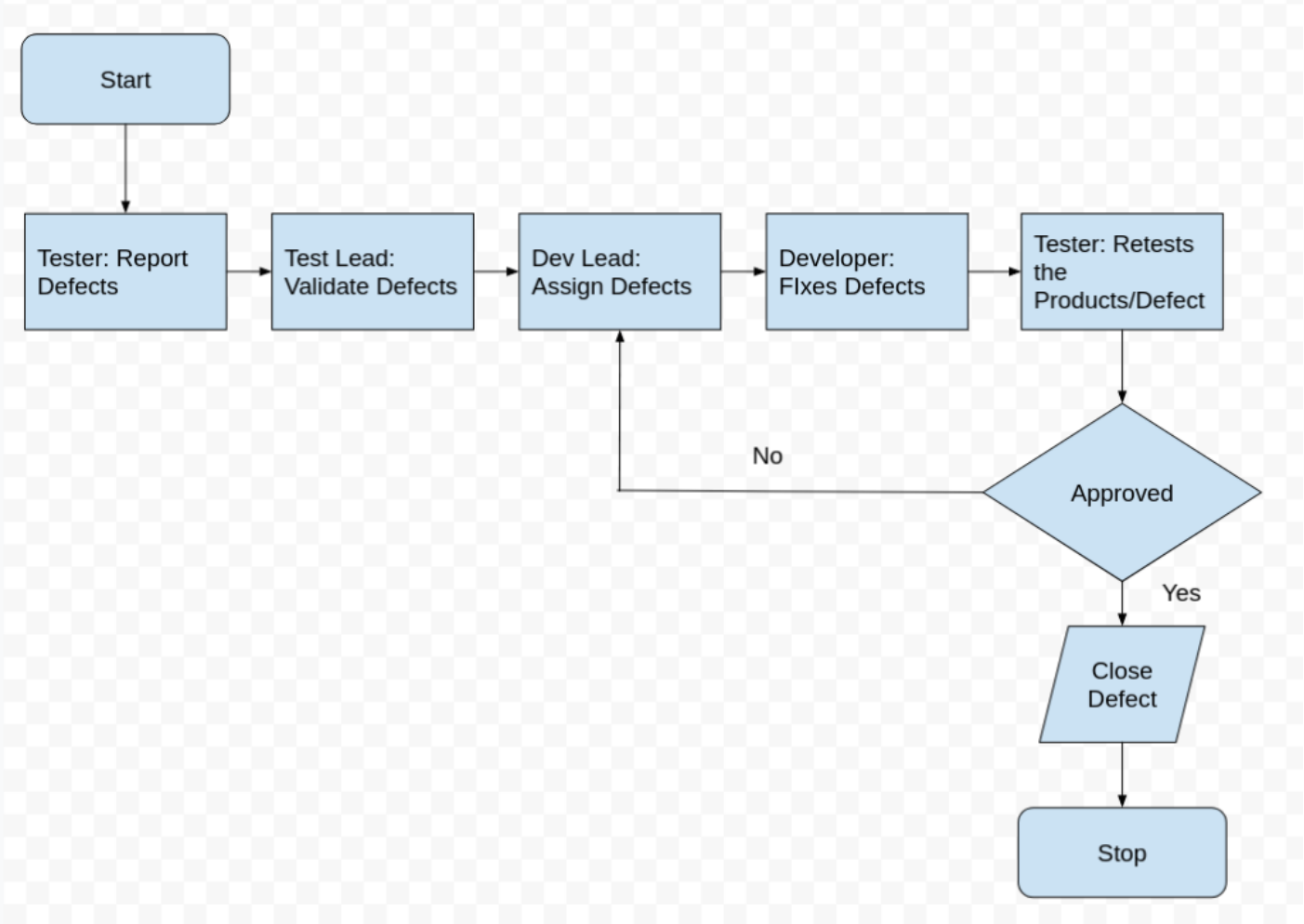


Fig: defect tracking & reporting flowchart

Defect Severity

Defect found during the Testing will be categorized according to the bug-reporting tool and the categories are:

Severity	Impact	Color Code
1 (Critical)	• This bug is critical enough to crash the system, cause file corruption, or cause potential data loss • It causes an abnormal return to the operating system (crash or a system failure message appears). • It causes the application to hang and requires rebooting the system.	#d40000
2 (High)	• It causes a lack of vital program functionality with a workaround.	#ff6600
3 (Medium)	• This Bug will degrade the quality of the system. However, there is an intelligent workaround for achieving the desired functionality - for example through another screen. • This bug prevents other areas of the product from being tested. However other areas can be independently tested.	#ffd42a
4 (Low)	• There is an insufficient or unclear error message, which has a minimum impact on product use.	#ffeeaa
5 (Cosmetic)	• There is an insufficient or unclear error message that has no impact on product use.	#ffaaaa

References:

https://www.softwaretestinghelp.com/defect-reporting-habits/

Bug Reporting Template

Who can report Defect/Bug?

Anyone came across the bug can report bug/defect and put that on the Project/Product Backlog.

After reporting the Bug QA team can organize the Triage meeting to set the priority and severity of that bug. Development Team lead, QA Team lead and Project Manager decide in which sprint that bug will be fix or that bug needs to be hotfix.

Note: If any team member other than QA found the bug/defect they have to assign that bug/defect to the QA of their project to verify whether that is bug/defect or not.

Template

Bug Title

A bug title is read more frequently than any other part of the bug report and should be clear and concise.

Everybody should be able to determine what the bug is without having to open the entire report.

Description

A well-defined bug description allows for an easy understanding of the bug and what happens.

Platform/Environment/Version/Context

It is also essential to include what operating system, browser configurations, and versions are being utilized when the bug is present.

Roles

Mention the in which user the bug has been replicated. (User roles)

Step to Reproduce

This provides clear instructions on how the bug can be reproduced by your developers.

Expected Results

Mention what is the expected results

Actual Results

Mention what is actually happening currently in that bug

Screenshot / Attachments / Screen Recording

Include a clear screenshot at the point of failure and when the bug occurred.

Example

Currently Playing shows data after flashing the ISO

Details

Context

All

Roles

- Have Permission to Flash Device
- Network Manager

Step to Reproduce

- Flash the Device which is already listed in the Salt Accepted list and working fine. which has Campaign playing
- Connect to the Internet
- Device now listed in the Denied Group
- Accept the Device in Denied Group
- Navigate to the Network Page
- Verify the Data on the Currently Playing & Last Sent

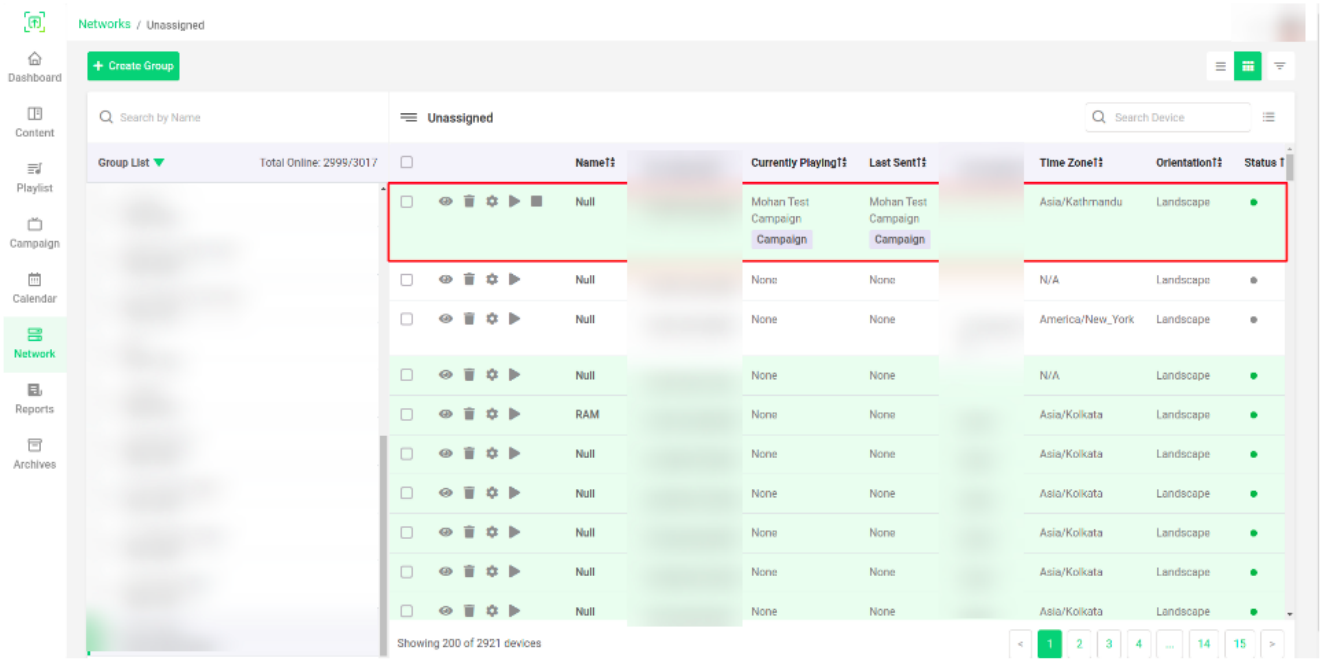
Expected Results

The currently playing, last sent data must be shown correctly in the UI

Actual Results

Previous data still remains on the currently playing after flashing the device

Attachments



Edited 4 days ago by Kishan Bhatta

Fig: Bug Example