JIRA Workflows

To handle issue and project tracking we use JIRA, which currently offers several different Issue Types when creating new tickets.

The expectations arising from each Issue Type are outlined below.

Issue Types

Additions

Issue Type	Reporter	Tracks	Deliverable(s)
Wish / New Feature	• Management • Developer	• Proposing new business logic	• New JIRA Tickets
Requirements	• Management • Developer	• Discussion of new features at a technical level	• New JIRA Tickets
Epic	• Developer	• Progress toward completing a high-level feature	• New Use Cases
Story	• Developer	• Introducing new Use Cases into the product • Progress toward the associated **Epic**	• Pull Request(s) • New Image(s) / Tag(s) • Documentation • New Test Case(s)

Alterations

Issue Type	Reporter	Tracks	Deliverable(s)
Improvement	• Developer • External Contributor	• Introducing new technologies or techniques into the platform • Increases in performance, usability, or maintainability ◦ Without adding or changing Use Cases	• Pull Request(s) • New Image(s) / Tag(s) • Updated documentation • Updated / new Test Case(s)
Bug	• Developer • External Contributor	• Divergences between expected Use Cases and product behavior	• Pull Request(s) • New Image(s) / Tag(s) • Updated documentation • Updated / new Test Case(s)

Accounting

Issue Type	Reporter	Tracks	Deliverable(s)

Comment	• Management • Developer • External Contributor	1. new sites / groups wishing to utilize the NDS Labs platform 2. similar technologies that we might look at for reference 3. new or existing technologies that might be leveraged 4. feedback-driven tasks	• New JIRA Tickets • Documentation
Processing Request	• Management • Developer • External Contributor	1. projects (via Account Creation Workflow) 2. service specs (via Pull Requests made to ndslabs-specs) 3. process-driven tasks	• New JIRA Tickets • Documentation • Modifications to etcd
Task	• Management • Developer	1. events requiring special attention (hackathon, conference, demo, etc) 2. externally-driven tasks	• New JIRA Tickets • Documentation
Sub-Task	• Management • Developer	• Progress towards the associated **Task** ticket • a small piece of technical work ◦ not driven by a new use case	• Documentation
Technical Task	• Management • Developer	• Progress towards the associated **Task** ticket • a small piece of outreach or non-technical work / discussion ◦ not driven by a new use case	• Documentation

Workflows

Addition Workflows

These issue types outline additions to the code base, in the form of new features (use cases).

Issue Types Used:

- **Wish** / **New Feature**: a high-level non-technical description of desired business logic
- **Requirement**: a "discussion" ticket describing a new feature that needs more of its technical description fleshed out
- **Epic**: a high-level technical description of desired software functionality or infrastructure containing multiple **Story** tickets
  - *Note*: **Epic** issues do not get explicitly added to Sprint
- **Story**: a use case describing an example usage of a small piece of newly desired functionality

General Relationship:

1. A **New Feature** or **Wish** ticket is filed describing the new feature at a high-level without mandating any particular technical specifications
2. A **Requirements** ticket is created if we feel that we do not have enough information to break the feature down into small pieces of technical work
3. A **Story** ticket is filed from the information resulting from the **Requirement** discussion
   a. If multiple **Story** tickets encompass the work needed, these tickets are grouped under an over-arching **Epic** ticket

New Feature Workflow

Reporter:

- Management
- Developer

Used to track:

- Introducing new business logic or new uses for existing logic into the product

Deliverables:

- New tickets describing the next steps necessary to enable the feature described

When a **Wish** or **New Feature** ticket is in the *Active Sprint:*

1. The ticket is marked `IN PROGRESS` and assigned to a developer, who conducts the meeting
2. The feature is broken down into a digestable number of enumerated use cases
3. If one or more use cases require more detail, a **Requirement** ticket is filed
4. The use cases are filed as **Story** tickets and associated to an **Epic**
5. The resulting **Story** tickets are then discussed at the next *Sprint Planning* meeting

Outstanding questions:

- When do I use a wish over a new feature?
  - Are **Wish** tickets requested by people external to NCSA?

- Does a **New Feature** have less detail than a **Wish**? Perhaps more detail?

Requirements Workflow

Reporter:

- Management
- Developer

Used to track:

- Discussion and creation of new **Story** / **Epic** tickets describing new product features at a technical level

Deliverables:

- A Confluence wiki page detailing the discussion held and the use cases generated from that discussion
- New **Epic** / **Story** tickets describing the steps necessary to enable the use cases described

When a **Requirement** ticket is in the *Active Sprint:*

1. The ticket is marked ***IN PROGRESS*** and assigned to a developer, who conducts the meeting
2. A meeting is created in Outlook to contact interested parties (i.e. NDS Labs Dev team, Nebula team, other NDS-affiliated software teams, etc.)
    a. The requirements are discussed with the development team and any interested parties
    b. Any information resulting from the discussion is filed into a Confluence wiki page
    c. The information from the Confluence page generates use cases
    d. If applicable, a new **Epic** is created to encompass the use cases presented
3. The use cases are filed as **Story** tickets and associated to an **Epic**
4. The ticket is marked ***IN REVIEW*** and assigned to another team member for review
5. The reviewer may make any changes or comments that they desire and discuss with the team
6. The ticket is marked ***RESOLVED*** *or* ***CLOSED***
7. The resulting **Epic** / **Story** tickets are then discussed at the next *Sprint Planning* meeting

Story Workflow

Reporter:

- Developer

Used to track:

- Introducing new Use Cases into the product
- Progress toward a particular **Epic** (i.e. a new technical feature consisting of multiple Use Cases)

Deliverables:

- **GitHub**: Pull Request(s)
- **DockerHub**: New Image(s) / Tag(s)
- **Confluence**: Documentation describing the technical aspects of how the platform fulfills the Use Case
- **Zephyr**: Test Case(s) demonstrating the use case fulfilled by the story
    - TODO: discover software for writing test plans

When a **Story** ticket is in the *Active Sprint:*

1. The ticket is marked ***IN PROGRESS*** and assigned to a developer (referred to hereafter as "the developer")
2. The developer does the work necessary to enable the use case described in the ticket
    a. Follow the general development workflows defined above
    b. Comment on the **Story** with links / updates to any deliverables that need to be reviewed / tested:
        i. Pull Requests
        ii. Docker Images
        iii. Documentation
        iv. New JIRA Tickets
3. The ticket is marked ***IN REVIEW*** and assigned to a tester (referred to hereafter as "the tester")
4. The tester reviews the deliverables of the **Story**:
    a. Review any related Pull Requests
    b. Review any Test Cases / Documentation provided
    c. Review any new JIRA tickets resulting from the work done
    d. Pull and run any new Docker images against the Test Cases provided
5. The tester needs to Accept, Reject, or Abort the review based on the results
    a. If the ticket does not contain sufficient information to decide whether or not the deliverables are acceptable, then the tester selects ***Review Aborted***
        i. The ticket is marked as ***OPEN*** and work is stopped on the ticket
        ii. The developer adds more detail to the ticket before continuing, for example:
            1. Test Case
            2. Passing Conditions
        iii. The developer then returns to #1 above and refines their deliverables
    b. If the deliverables are missing, incomplete, or in an untestable state, then the tester selects ***Review Rejected***
        i. The ticket is marked as ***IN PROGRESS*** and should then be assigned back to the developer
        ii. The developer then returns to #1 above and refines their deliverables

    c. If the deliverables are tested and in an acceptable form, then the tester selects ***Review Accepted***
        i. The ticket is marked as ***RESOLVED***
        ii. The tester continues the workflow below
6. The tester merges any outstanding Pull Requests related to this ticket
7. The developer switches back to **master** and syncs with upstream (to pull the new changes into their master branch)
8. If applicable, the developer builds and pushes a new "latest" Docker image for the API / UI incorporating the new changes
9. The developer selects ***CLOSE TICKET*** and the ticket is marked as ***CLOSED***

Alteration Workflows

These issue types outline modifications to existing features (use cases).

Improvement: a suggestion that might have a positive impact on the product without introducing new features (i.e. refactoring, rewriting, etc.)Issue Types Used:

- **Bug**: a previously completed use case or edge case that is malfunctioning according to its defined behavior

General Relationship:

1. A **Bug** or **Improvement** ticket is filed detailing a potential modification that will have a positive impact on the platform
2. If necessary, a **Requirement** ticket is filed to explore the ramifications of the changes

Improvement Workflow

Improvement tickets follow a workflow that resembles that of a **Story** ticket, with some slight modifications.

Reporter:

- Developer
- External Contributor (via GitHub "enhancement" issue)

Used to track:

- Introducing new technologies or techniques into the underlying platform
- Increases in performance, usability, or maintainability without adding or changing Use Cases

Deliverables:

- **GitHub**: Pull Request(s)
- **DockerHub**: New Image(s) / Tag(s)
- **Confluence**: Updated technical documentation that reflects any modifications to the platform
- **Zephyr**: Test Case(s) exercising the benefit introduced by this improvement
    - TODO: discover software for writing test plans

When an **Improvement** ticket is in the *Active Sprint:*

1. Ticket Status: Start Progress
2. Developer creates a new branch with a small prototype instance containing suggested improvement(s)
3. Developer weighs the pros / cons of this solution over the current one against the time it would take to fully implement and test the change
    a. If not desirable, the developer abandons the branch(es) containing these changes and marks the ticket as CLOSED
    b. If desirable, the developer completes the modifications on the branch
4. Once complete, developer gathers the necessary deliverables:
    a. **Confluence**: Create documentation and/or take note of technical details
    a. **GitHub**: Create Pull Request(s)
    b. **DockerHub**: Create and push a test image tagged with the name of the corresponding git branch
    c. **Zephyr**: Create new / update existing test cases relating to the modifications
5. Ticket Status: Review Ticket and assign to Tester
6. Tester reviews / tests any deliverables
    - **Confluence**: Review any relevant documentation or technical details
    - **GitHub**: Review related Pull Request(s)
    - **DockerHub**: Pull and run test image(s) against test cases
    - **Zephyr**: Run any new / updated test cases relating to the modifications
7. Tester merges any Pull Requests (if applicable)
8. Ticket Status: Resolve Ticket and assign back to Developer
9. Developer releases other deliverables themselves
    a. **Confluence**: Developer migrates any relevant documentation from Personal Space to "National Data Service" public space (if applicable)
    b. **GitHub**: Developer syncs with upstream changes (if applicable)
        i. git checkout master
        ii. git pull upstream master
        iii. git push origin master
    d. **DockerHub:** Developer builds and pushes new "latest" stable Docker images
    e. **GitHub**: Developer commits and pushes new build date upstream (if applicable)
10. Ticket Status: Close Ticket

Bug Workflow

Bug tickets follow a workflow that resembles that of a **Story** ticket, with some slight modifications.

Reporter:

- Developer
- External Contributor (via GitHub "bug" issue)

Used to track:

- Divergences between expected Use Cases and product behavior

Deliverables:

- **GitHub**: Pull Request(s)
- **DockerHub**: New Image(s) / Tag(s)
- **Confluence**: Updated technical documentation that reflects any modifications to the platform
- **Zephyr**: Updated / new Test Case(s) reflecting the expected behavior of the product
    - TODO: discover software for writing test plans

When a **Bug** ticket is in the *Active Sprint:*

1. Ticket Status: Start Progress
    a. aside: consider creating a new status for "Verification" stage of Bug tickets?
2. Developer reviews affected Use Case(s) and verifies behavioral divergence
    a. Developer examines validity and may suggest modifications to the Use Case
    b. Developer determines where the bug stems from in the source
    c. Developer devises one or more ways to address the bug in question
    d. Developer selects the "best" option according to their judgement given the circumstances
3. Developer fixes product behavior to match expected Use Case
4. Once complete, developer gathers any necessary deliverables:
    a. **Confluence**: Create documentation and/or take note of technical details
    b. **GitHub**: Create Pull Request(s)
    c. **DockerHub**: Create and push a test image tagged with the name of the corresponding git branch
    d. **Zephyr**: Create new / update existing test cases relating to the modifications
5. Ticket Status: Review Ticket and assign to Tester
6. Tester reviews / tests any deliverables
    a. **Confluence**: Review any relevant documentation or technical details
    a. **GitHub**: Review related Pull Request(s)
    b. **DockerHub**: Pull and run test image(s) against test cases
    c. **Zephyr**: Run any new / updated test cases relating to the modifications
7. Tester merges any Pull Requests (if applicable)
8. Ticket Status: Resolve Ticket and assign back to Developer
9. Developer releases other deliverables themselves
    a. **Confluence**: Developer migrates any relevant documentation from Personal Space to "National Data Service" public space (if applicable)
    b. **GitHub**: Developer first syncs with upstream changes (if applicable)
        i. git checkout master
        ii. git pull upstream master
        iii. git push origin master
    c. **DockerHub:** Developer builds and pushes new "latest" stable Docker images
    d. **GitHub**: Developer commits and pushes new build date upstream (if applicable)
10. Ticket Status: Close Ticket

Accounting Workflows

These issue types outline non-development tasks or support requests related to the platform.

- **Comment**: track miscellaneous information / feedback / requests that do not match other issue typesIssue Types Used:
- **Processing Request**: track the creation of new entities in production instance of NDS Labs
- **Task**: work that is not driven by a new use case. Contains zero or more **Sub-Issue** tickets
    - **Technical Task**: a small piece of technical work that is not driven by a new use case
    - **Sub-Task**: a small piece of outreach or non-technical work / discussion that is not driven by a new use case

General Relationship:

1. A **Comment** / **Processing Request** / **Task** ticket is filed to track work that must be tracked
    a. For these tasks, it is likely that you will not need to make any actual modifications to the code
2. Larger **Task** tickets can be broken up into a series of **Technical Task** and **Sub-Task** tickets
3. If necessary, a **Requirement** ticket is filed to explore the nature and limits of the support granted by this ticket

Comment Workflow

Reporter:

- Management
- Developer

- External Contributor

Used to Track:

1. new sites / groups / contacts wishing to utilize the NDS Labs platform (i.e. Odum, TACC, SDSC, etc.)
2. similar technologies that we might look at for reference (i.e. JujuCharms, ProfitBricks, etc.)
3. new or existing technologies that might be leveraged to further NDS Labs
4. any other feedback-driven tasks that require explicit work to be done

Deliverables:

- If necessary, a **Requirement** ticket is filed to reflect on the meaning and validity of the comment

When a **Comment** ticket is in the *Active Sprint:*

- Do these issues get added to sprint?

Request Workflow

Reporter:

- Management
- Developer
- External Contributor

Used To Track:

1. projects (via Account Creation Workflow)
2. service specs (via Pull Requests made to ndslabs-specs)
3. any other process-driven tasks that require explicit work to be done

Deliverables:

- If necessary, a **Requirement** ticket is filed to discuss any further changes that might be necessary to process this request

When a **Processing Request** ticket is in the *Active Sprint:*

- Do these Issues get added to Sprint?

Task Workflow

Reporter:

- Management
- Developer

Used to Track:

1. events requiring special attention (i.e. hackathon, developer tutorial, etc.)
2. any other externally-driven tasks that require explicit work to be done

Deliverables:

- If necessary, a **Requirement** ticket is filed to determine any necessary hardware/software requirements prior to supporting the event

When a **Task / Sub-Task** / **Technical Task** ticket is in the *Active Sprint:*

- Do these Issues get added to Sprint?