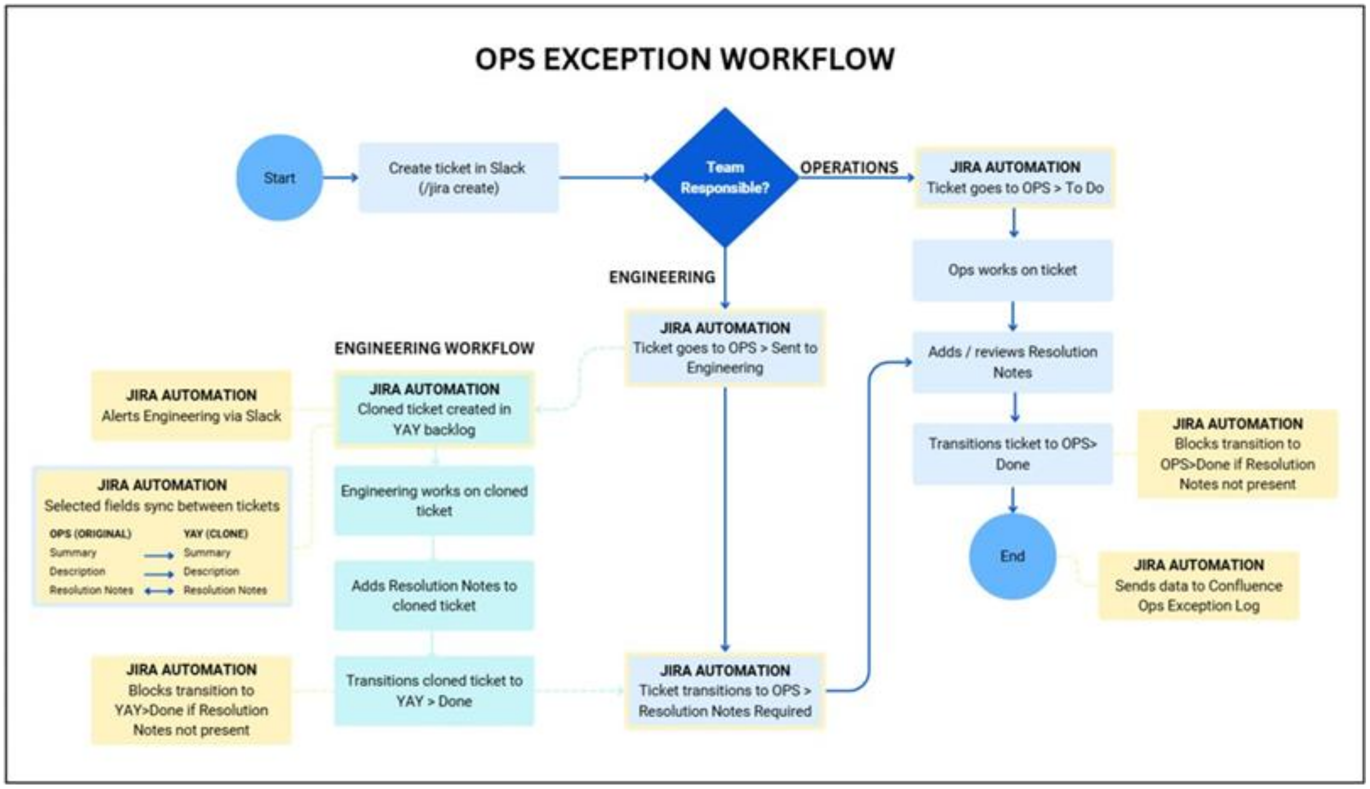
How-To: Ops Exceptions Workflow

This workflow ensures every exception reported by the Operations team is captured, handled, and logged into Confluence.

Figure 1: The movement of tickets between the Operations and Engineering team, and the accompanying Jira automations.


OPS EXCEPTION WORKFLOW
Start
Create ticket in Slack (/jira create)
Team Responsible?
OPERATIONS
JIRA AUTOMATION Ticket goes to OPS > To Do
Ops works on ticket
Adds / reviews Resolution Notes
Transitions ticket to OPS> Done
JIRA AUTOMATION Blocks transition to OPS>Done if Resolution Notes not present
End
JIRA AUTOMATION Sends data to Confluence Ops Exception Log
ENGINEERING
JIRA AUTOMATION Ticket goes to OPS > Sent to Engineering
ENGINEERING WORKFLOW
JIRA AUTOMATION Alerts Engineering via Slack
JIRA AUTOMATION Cloned ticket created in YAY backlog
JIRA AUTOMATION Selected fields sync between tickets
OPS (ORIGINAL) Summary Description Resolution Notes
YAY (CLONE) Summary Description Resolution Notes
Engineering works on cloned ticket
Adds Resolution Notes to cloned ticket
JIRA AUTOMATION Blocks transition to YAY>Done if Resolution Notes not present
Transitions cloned ticket to YAY > Done
JIRA AUTOMATION Ticket transitions to OPS > Resolution Notes Required

Part 1 - Ticket Creation

If there is an issue that needs to be investigated and resolved by an Operations team member or by the Engineering team, then a ticket should be created in Slack and sent to the **Ops Issues & Exceptions (OPS)** project in Jira.

To create a ticket:

1. Go to the Slack channel #ops-exceptions.
2. Type **/jira create** to open the ticket creation template.
3. Complete the fields. (See Table 1 for guidance.)
4. Click **Submit**.

If you want to add addition details that were not captured in the Slack template (screenshots, etc.), update the *Description* field in the ticket on the OPS board. The field is synced and will automatically update with the cloned ticket.

Table 1: Template fields in Slack when creating an Ops Exception ticket

Field	How to populate
Project	Ops Issues & Exceptions *(leave as default)*
Issue type	Ops Exception *(leave as default)*
Summary (mandatory)	Summarize the issue in one line.
Description	Add a detailed description of the problem. What were you doing and when, what did you expect to happen, and what actually did happen?
Exception Type (select one, mandatory)	• Platform / app issue • Payments and transactions • Customer experience • Event operations • Internal ops and admin • Other (needs triage)
Team Responsible (select one, mandatory)	• Engineering *(if technical issue other than permissions)* • Operations
Context Tags (select as many as needed, optional)	• Platform - functional issue • Platform - UI issue • Platform – event / venue visibility • Platform - integrations • Platform – outage / latency • Platform - mobile app • Ops - scheduling • Ops - ticketing • Ops – supplies / logistics • Ops – access / permissions • Ops - reporting • Finance - payment processing • Finance - refunds • Finance - taxes • Marketing – promotions / discounts • CX – email / notifications
Priority (select one, optional)	• General *(default)* • Major • Critical • Blocker

Part 2 - Ticket Triaging and Resolution

The next part of the workflow depends upon how you populated the *Team Responsible* field.

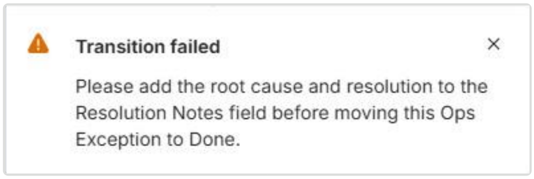
[Team Responsible = Engineering] workflow

If the ticket indicates the team responsible is *Engineering*:

What happens on the Engineering side:

1. The ticket is cloned automatically and lands in the backlog of the **Engineering Kanban (YAY)** project in Jira.
2. The engineering team receives an alert in the #ops-exceptions-eng-triage channel that a new *Ops Exception* ticket has been created.
3. Engineers can either click the link in Slack directly to open the ticket, or filter the backlog using the custom filter *Triage Ops Exceptions* to find the clone and move it to the Kanban board.
4. The engineers work in the cloned ticket and update as work progresses.
5. When complete, they add the root cause and resolution to the *Resolution Notes* field and transition the cloned ticket to **Done**.

An Ops Exception ticket cannot be transitioned to **Done** if the *Resolution Notes* field is empty. If you try to do so, you will receive this prompt:


Transition failed

Please add the root cause and resolution to the Resolution Notes field before moving this Ops Exception to Done.

What happens on the Operations side:

1. The original ticket lands in the **OPS board > Sent to Engineering** column.
2. The original ticket is linked to the cloned ticket, so Ops can easily follow progress in the cloned ticket if desired. The *Resolution Notes* field is synced, so any updates made to that field will be reflected in both tickets.
3. When the engineers move the cloned ticket to **YAY board > Done**, the original ticket moves to the **OPS board > Resolution Notes Required** column.
4. The original reporter will review the *Resolution Notes* field in the ticket:
   - If the root cause and resolution notes are clear, they transition the ticket to **Done**.
   - If the notes need updating, they do so then transition the ticket to **Done**.

Comments do not sync between tickets to avoid excessive noise and duplicate tagging, and they are not included when details are logged to Confluence.

If an update made in comments is important for long-term tracking or root cause analysis, it must also be added to the *Resolution Notes* field.

[Team Responsible = Operations] workflow

If the ticket indicates the team responsible is *Operations*:

What happens on the Engineering side:

Nothing.

What happens on the Operations side:

1. The ticket lands in the **OPS board > To Do** column.
2. The team assigns the appropriate member to work on the ticket and keep it updated as work progresses.
3. When complete, they add the root cause and resolution to the *Resolution Notes* field and transition the ticket to **Done**.

Part 3 - Logging Exceptions in Confluence

Whenever a ticket reaches **OPS Board > Done**, an automation publishes detail to the *Ops Exception Log* in the Operations Team space, including:

Table 2: Fields in the Ops Exception Log in Confluence

Field	Description
Ticket Link	Direct Jira issue link
Exception Type	Category for the type of error/exception
Context Tags	Tags applied for grouping / searching in Confluence
Summary	Short title of the ticket
Description	Full text entered when creating the ticket
Resolution Notes	Summary of root cause, actions taken, and resolution
Created	Date ticket was created
Resolved	Date ticket transitioned to Done
Reporter	Original reporter (owner of ticket)

The log that is built through this workflow will provide a single source of truth for recurring issues, making it easier to spot patterns, share knowledge, and prevent the same problems from happening again.