Jira and Azure DevOps Integration Documentation

# 1. How Jira Tasks are Linked to Azure DevOps

Jira and Azure DevOps are integrated using the official Atlassian 'Azure DevOps for Jira' app. This integration allows Jira to detect development activities such as commits, branches, pull requests, and deployments performed in Azure DevOps. The link is established by matching Jira issue keys (e.g., LVDP-101) mentioned in development artifacts with the Jira project configuration.

Jira detects and links development activity through:

* - Commit messages containing Jira issue keys (e.g., 'LVDP-101: Initial commit')
* - Pull request titles or descriptions containing the Jira issue key
* - Branch names containing the Jira issue key

Once detected, this information appears in the Jira issue's Development panel, showing linked commits, branches, and pull requests. This linkage ensures traceability of code changes back to specific business requirements or tasks.

# 2. Planned Jira Automation Rules

## Rule 1: Move Jira Issue to "In Progress" When a Commit is Made

This rule ensures that when a developer starts working on a task and pushes a commit that includes the Jira issue key, the Jira issue status automatically transitions from 'To Do' to 'In Progress'.

* Trigger: Commit Created
* Condition: Issue status = To Do (optional for control)
* Action: Transition issue to 'In Progress'
* Note: The Jira key must be present in the commit message for this rule to trigger.

## Rule 2: Move Jira Issue to "Done" When Pull Request is Merged

This rule ensures that once a feature is completed and the pull request is merged in Azure DevOps, the corresponding Jira issue is automatically closed.

* Trigger: Pull Request Merged
* Condition: Issue status ≠ Done (optional)
* Action: Transition issue to 'Done'
* Optional Action: Add comment such as 'Completing Pull Request and the associated Jira issue.'
* Note: The Jira key must be present in the PR title or description for this rule to trigger.

# 3. Optional: Using Two-Way Synchronization Tools (e.g., Unito, Exalate)

While the official Azure DevOps for Jira integration is one-way (from ADO to Jira), enterprises that require full synchronization between Azure DevOps work items and Jira issues often use third-party integration platforms. These tools support bi-directional sync, which can be critical for organizations where different teams use different platforms.

## A. Unito

Unito is a no-code integration platform that allows real-time, two-way sync between tools like Jira and Azure DevOps. It can sync fields such as title, description, assignee, comments, labels, status, and due dates.

* Key features include:
* - Two-way syncing of Jira issues and ADO work items
* - Visual flow builder for mapping fields
* - Sync filters for selective syncing

## Use Case with Unito

Scenario: Product Manager creates a Jira ticket (e.g., LVDP-110) in the planning phase.

1. Workflow:
2. Unito detects the new Jira ticket and automatically creates a linked work item (e.g., user story or task) in Azure DevOps.
3. The developer picks up the ADO task and begins implementation. As they create a pull request, they can link it to the ADO work item, which is already synced with the Jira ticket.
4. Any status update, comment, or assignment change on either side (Jira or ADO) is mirrored across both platforms.
5. Once the PR is merged, the ADO work item is marked as complete, and Unito syncs this change back to Jira, transitioning the Jira issue to 'Done'.

## B. Exalate

Exalate is a flexible synchronization solution built for more complex enterprise needs. It supports advanced scripting and can be customized for selective field syncs and transformation rules.

* Key features include:
* - Fully customizable sync rules using scripting
* - Secure and decentralized architecture
* - Sync attachments, comments, and custom fields

## Use Case with Exalate

Scenario: Development team uses Azure DevOps while the business/PMO uses Jira.

1. Workflow:
2. Jira issue is created (e.g., LVDP-150) with business requirements.
3. Exalate triggers a sync based on configured rules and creates a work item in Azure DevOps with mapped fields (title, description, priority, etc.).
4. Developer starts work in ADO. When they commit code or create a pull request, they tag the ADO work item.
5. Because the ADO item is already linked to the Jira issue via Exalate, this PR association is implicitly traceable in Jira.
6. Exalate updates the Jira issue status, comments, and fields automatically based on actions taken in ADO, such as PR merges or status changes.

## Considerations for Using Two-Way Tools

Before adopting Unito, Exalate, or similar tools, consider:

* - Licensing and cost (enterprise tools often have tiered pricing)
* - Configuration complexity and ongoing maintenance
* - Whether full two-way sync is necessary, or if one-way is sufficient
* - Access and security policies (especially if dealing with external vendors)

## Summary of Benefits

* - Developers never need to manually create matching work items in ADO.
* - PMs and business teams retain full visibility in Jira, while developers stay focused in ADO.
* - Pull requests in ADO can still be linked to ADO work items, which are already synced with Jira issues.
* - Status alignment between tools is automated, reducing human error and ensuring real-time reporting.