DevInfo v0.10 - Vendor Implementation Guidance

This document explains how a third party can use the new development information API to integrate their development tool with JIRA. Connect is the mechanism by which Atlassian cloud products are extended, information about working with Connect can be found on the Atlassian developer documentation site - https://developer.atlassian.com/.

Connect

This document will be easier to understand if you understand Atlassian connect. The main features you need to be aware of are:

- Connect is the framework that allows third parties to extend Atlassian cloud products.
- An 'app' is the term used to refer to a third party implementation using the connect framework to extend a product.
- Connect uses JWT for authentication.
- An app defines the extension points it will use in JIRA through the 'descriptor' which is a json document.
- In the descriptor there are a list of 'modules' which define the specific extension points or features being used. For example, this page shows the configuration for a keyboard shortcut https://developer.atlassian.com/cloud/jira/platform/modules/keyboard-shortcut/.

Getting Started with Connect guide

This section outlines how to set up a Jira Cloud development environment to build and test your app on the Connect framework.

Step 1: Create a free development instance of Atlassian Cloud (comes with Jira Software, Jira Core, Jira Service Desk, and Confluence). go.atlassian.com/cloud-dev

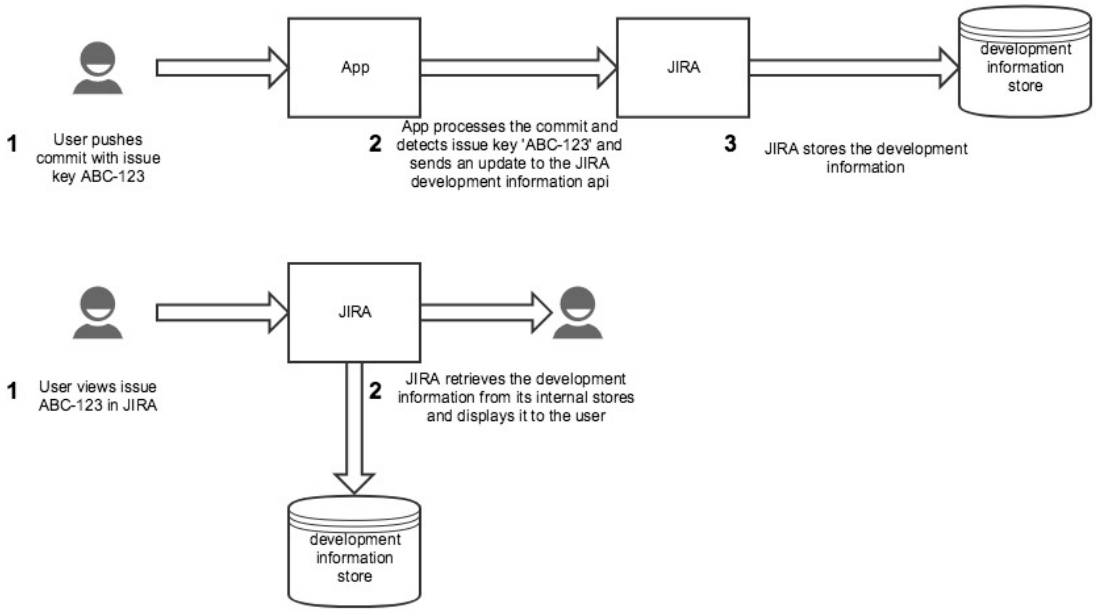
Step 2: Follow this getting started guide

Additional Info

- The 'app descriptor' json file is where you will use the new module we have developed for sending development information, called 'jiraDevelopmentTool'.
- Then, upload the descriptor file on your Jira cloud developer instance by going to Settings > Add-ons > Manage add-ons. The descriptor file will need to be hosted and accessible over the web.
- Then application will be able to use Devinfo API to upload development information to Jira. When you go to your board and select an issue that has development information associated with it, you should see information displayed.

Overall flow for the integration

This diagram shows the overall flow of what happens when a commit for an issue is created in a development tool and then the issue viewed in JIRA.



Defining the app

Your Connect app will need to define the 'jiraDevelopmentTool' module. In addition to the standard module key this module will require the following fields:

Field		Description
Application	Internationalised Property	The application that is providing the development information. Development information from the same application will be grouped together for display.
Name	Internationalised Property	The name of the application.
Url	Url	A url to the application.
logoUrl	Url	An icon for the application that will be displayed in the JIRA ui.
Capabilities	List of Strings	Information for a user about what information this app will support, this is a List which can contain "commit", "branch" and "pullrequest" values. This field is purely informational. Capability information will be displayed in UI to describe the functionality of this development tool.

If **jiraDevelopmentTool** module is not specified, Devinfo API will not be accessible and all request will fail with 403 response.

Authentication

During the install process JIRA will send you a secret that you must use to make JWT authenticated requests to JIRA. See https://developer.atlassian.com/cloud/jira/platform/security-overview/ for more information on security in Atlassian Connect.

Using the development information API

The below Swagger specification defines the API contract that can be used to push development information to JIRA. There are also a DELETE and GET operations for data removal and retrieval

Click here to expand...

The main API that should be used is `/rest/devifo/0.10/` which can be used to supply Commits, Branches and Pull Requests to JIRA. Data is grouped by repositories, and there is some cross referencing between the entities. Particularly to match Branches and Pull Requests (through the branch url and pull request sourceBranchUrl).

The expected usage is similar to a webhook, as commits are pushed or pull requests are created you can invoke this API and JIRA will capture the relevant information. The app will need to scan the commit message, pull request title or other fields to determine which issue should be updated and then send the request to JIRA to attach the dev information.

If an entity with the same `id` value is sent then it will override the existing data for that `id` (if request data is more fresh than existing, which is determined by updateSequenceId parameter of individual entity, skipped otherwise).

Issue Transitions

JIRA workflows can be configured to transition issues when development activity occurs, for example you could move an issue to 'in progress' when a commit is created. JIRA will automatically deduce the type of change required when you POST dev information to the API. I.e. if a Commit is pushed a CommitCreated event will fire. If a PR's state changes from open to merged a 'PR Merged' event will fire.

To disable this processing there is a global flag on the request preventTransitions which can be set to false to disable these transitions. The typical scenario for ignoring transitions is performing initial 'load' of historical dev information to JIRA.

The scenarios where JIRA will perform a transition are:

Action	Trigger that will be fired
Commit in request	Commit created.
Branch in request	Branch created.
Pull Request that is not already stored in JIRA	Pull request created. Note that this fire even if the pull request is not in an 'open' state, if the first status of a pull request is 'merged' we will fire both a merged and created event

Pull Request in request with status open and existing pull request with status declined or merged	Pull request reopened.
Pull Request in request with status merged and existing pull request with status other than merged	Pull request merged.
Pull Request in request with status declined and existing pull request with status other than declined	Pull request declined.

The scenarios where JIRA will not perform a transition are:

Action	Field	Condition	Description
Commit in request	authorTimestamp	Older than two weeks	
Commit in request	id	Existing commit with this id	We will lookup our persistent store, if the commit already exists with this id then no event is fired
Branch in request	id	Existing branch with this id	We will lookup our persistent store, if the branch already exists with this id then no event is fired
Pull request in request	lastUpdate	Older than two weeks	

Smart Commits

If preventTransitions is false or not supplied and the commit message contains a smart commit command, JIRA will process this command and perform the action specified. See https://confluence.atlassian.com/bitbucket/processing-jira-software-issues-with-smart-commit-messages-298979931.html for more information on smart commits.

> For smart commit case Issue ID from commit message will be used to make a commit-issue association. Issues list property will be ignored. (It is used for displaying commits on issues in Jira UI and Issue transitions triggering)

Sample request.

This request shows a sample json payload to create few commits, branch and pull request against some issues in JIRA.

Click here to expand...

```
{
  "preventTransitions": false,
  "repositories": [
    {
      "url": "https://example.com/repo1",
      "name": "repo1",
      "avatar": "https://d301sr5gafysq2.cloudfront.net/38c40bdeb954/img
/repo-avatars/default.svg",
      "avatarDescription": "Something witty",
      "description": "The repo1 repository",
      "forkOf": null,
      "commits": [
        {
          "url": "http://example.com/path/to/commit1",
          "hash": "hash1",
          "displayId": "displayId1",
          "authorTimestamp": "2016-10-31T23:27:25+00:00",
          "author": {
            "name": "Josie Smith",
```

```
            "avatar": "http://example.com/avatar/josie.png",
            "url": "http://example.com/profile/josie",
            "email": "josie@example.com",
            "username": "jsmith"
          },
          "fileCount": 0,
          "flags": [],
          "message": "First commit",
          "files": [],
          "id": "dadd6957-c1f3-4341-af6b-f53c249e42e7",
          "issueKeys": [
            "TA-4"
          ],
          "updateSequenceId": 1
        },
        {
          "url": "http://example.com/path/to/commit2",
          "hash": "hash2",
          "displayId": "displayId2",
          "authorTimestamp": "2017-10-31T23:27:25+00:00",
          "author": {
            "name": "Josie Smith",
            "avatar": "http://example.com/avatar/josie.png",
            "url": "http://example.com/profile/josie",
            "email": "josie@example.com",
            "username": "jsmith"
          },
          "fileCount": 0,
          "flags": [],
          "message": "Second commit",
          "files": [],
          "id": "97d19a23-cb13-4d96-837c-ee0b3e4f7e78",
          "issueKeys": [
            "TA-3",
            "TA-4"
          ],
          "updateSequenceId": 1
        }
      ],
      "branches": [
        {
          "url": "https://example.com/repo1/branch1",
          "name": "branch1",
          "createPullRequestUrl": "https://blah.example.com/createpr?
branch=branch1",
          "lastCommit": {
            "url": "http://example.com/path/to/commit2",
            "hash": "hash2",
            "displayId": "displayId2",
            "authorTimestamp": "2017-10-31T23:27:25+00:00",
            "author": {
              "name": "Josie Smith",
              "avatar": "http://example.com/avatar/josie.png",
```

```
            "url": "http://example.com/profile/josie",
            "email": "josie@example.com",
            "username": "jsmith"
          },
          "fileCount": 0,
          "flags": [],
          "message": "Second commit",
          "files": [],
          "id": "97d19a23-cb13-4d96-837c-ee0b3e4f7e78",
          "issueKeys": [
            "TA-3",
            "TA-4"
          ],
          "updateSequenceId": 1
        },
        "id": "78adffc4-6a36-4dd5-b6c5-cbcd4eed21ac",
        "issueKeys": [
          "TA-3",
          "TA-4"
        ],
        "updateSequenceId": 1
      }
    ],
    "pullRequests": [
      {
        "url": "https://example.com/repo1/branch1",
        "displayId": "pr1 display",
        "author": {
          "name": "Josie Smith",
          "avatar": "http://example.com/avatar/josie.png",
          "url": "http://example.com/profile/josie",
          "email": "josie@example.com",
          "username": "jsmith"
        },
        "title": "TA-4 Sample PR",
        "commentCount": 42,
        "sourceBranch": "https://example.com/repo1/branch1",
        "destinationBranch": "https://example.com/repo1/master",
        "reviewers": [
          {
            "name": "Claira Bear",
            "avatar": "http://example.com/avatar/claira.png",
            "url": "http://example.com/profile/claira",
            "approvalStatus": "UNAPPROVED"
          }
        ],
        "status": "OPEN",
        "lastUpdate": "2017-12-30T23:18:22+00:00",
        "id": "881f5377-39f2-48d6-bcdc-330d5f660478",
        "issueKeys": [
          "TA-4"
        ],
        "updateSequenceId": 1
```

```
          }
        ],
        "id": "d2e56c5c-5cae-4852-9fc6-4f25abee72e6",
        "updateSequenceId": 1
      }
    ],
    "properties": {
      "accountId": "1234"
    }
  }
```