Jira Development Information API - Vendor Implementation Guidance

> **Important**
> Please note this information is to provide early guidance on upcoming functionality that will allow third parties to send development information to be displayed in JIRA. This is not a final specification and subject to change slightly in the future.

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/.

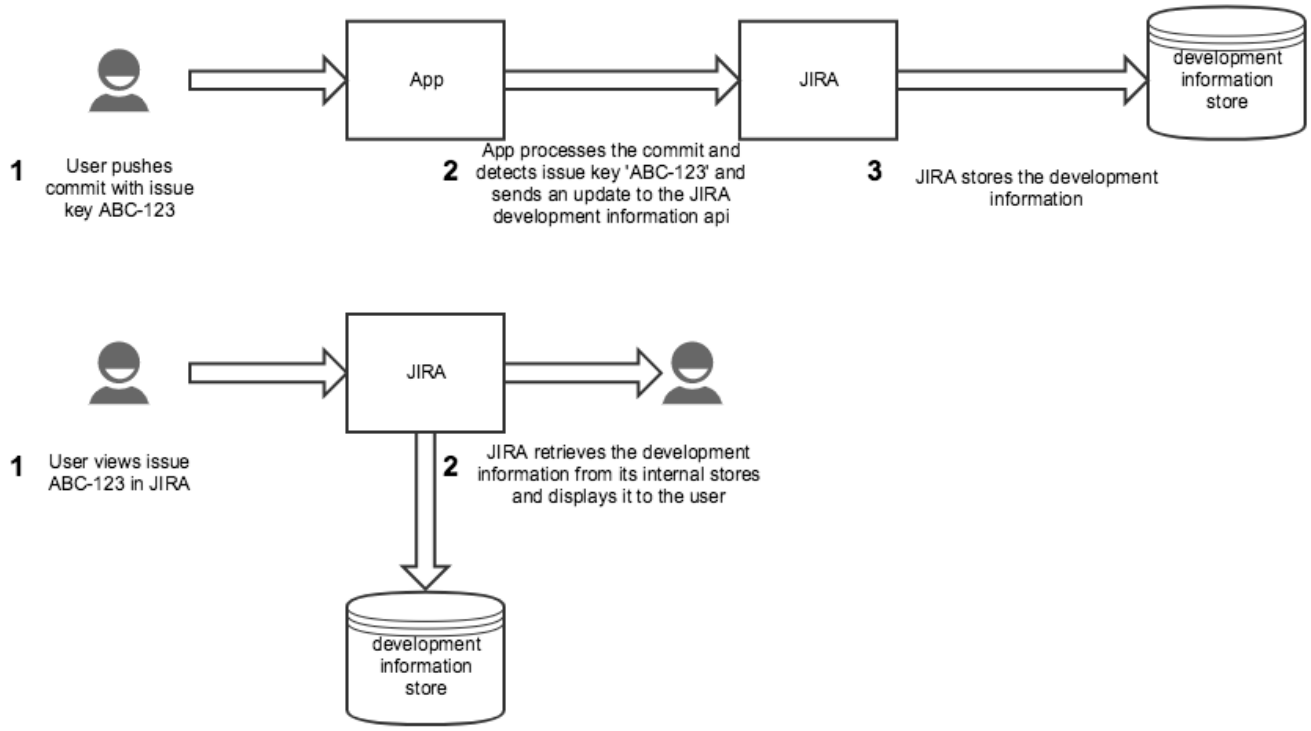
Connect

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/.

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 'devIntegration' module. In addition to the standard module key this module will require the following fields:

Field	Description
Application	The application that is providing the development information. Development information from the same application will be grouped together for display.
Name	The name of the application.
Url	A url to the application.
Icon	An icon for the application that will be displayed in the JIRA ui.
Capabilities	Information for a user about what information this app will support, this is a List which can contain COMMIT, BRANCH and PULL_REQUEST.

Except for Capabilities all these fields support internationalisation as per the connect specification. The Capabilities will be translated by JIRA and converted to a user friendly wording.

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 is also a DELETE operation to remove all data.

Click here to expand...

```
{
  "swagger": "2.0",
  "info": {
    "description": "Start the hole filling for repositories",
    "version": "1.0",
    "title": "DSS Commit Processor API"
  },
  "host": "localhost",
  "basePath": "/api",
  "schemes": [
    "https"
  ],
  "consumes": [
    "application/json"
  ],
  "produces": [
    "application/json"
  ],
  "paths": {
```

```
    "/external/devInformation/{issuekey}": {
      "post": {
        "summary": "Add dev information for the specified provider on
this issue. This operation will add these values but will not remove any
data, any data with the same idwill be overwritten",
        "description": "",
        "operationId": "setDevInformation",
        "consumes": [
          "application/json"
        ],
        "produces": [
          "application/json"
        ],
        "parameters": [
          {
            "in": "body",
            "name": "body",
            "required": true,
            "schema": {
              "$ref": "#/definitions/DevInformation"
            }
          },
          {
            "name": "issuekey",
            "in": "path",
            "description": "The issue key against which the development
information is being stored",
            "required": true,
            "type": "string"
          }
        ],
        "responses": {
          "200": {
            "description": "Dev information stored."
          }
        }
      },
      "delete": {
        "summary": "Remove all dev information on this issue. ",
        "description": "",
        "operationId": "deleteDevInformation",
        "consumes": [
          "application/json"
        ],
        "produces": [
          "application/json"
        ],
        "parameters": [
          {
            "name": "issuekey",
            "in": "path",
            "description": "The issue key against which the development
information is being stored",
```

```
            "required": true,
            "type": "string"
          }
        ],
        "responses": {
          "200": {
            "description": "Dev information stored."
          }
        }
      }
    }
  },
  "definitions": {
    "PullRequest": {
      "type": "object",
      "required": [
        "author",
        "displayId",
        "id",
        "lastUpdate",
        "repositoryId",
        "sourceBranch",
        "status",
        "title",
        "url"
      ],
      "properties": {
        "status": {
          "type": "string",
          "description": "The status of the pull request. Priority
applies between these states with preference given in the order OPEN,
MERGED, DECLINED, UNKNOWN",
          "enum": [
            "OPEN",
            "MERGED",
            "DECLINED",
            "UNKNOWN"
          ]
        },
        "repositoryId": {
          "type": "string",
          "description": "The id of the repository for this pull
request, will be used to lookup the repository details when displaying
this pull request"
        },
        "author": {
          "readOnly": true,
          "$ref": "#/definitions/Author"
        },
        "displayId": {
          "type": "string",
          "description": "A shortened identifier for the pull request
suitable for display"
```

```
        },
        "title": {
          "type": "string"
        },
        "url": {
          "type": "string",
          "description": "The url for this pull request"
        },
        "commentCount": {
          "type": "integer",
          "format": "int32"
        },
        "sourceBranch": {
          "type": "string",
          "description": "The id of the source branch of this PR. This
is used to match this PR against the branch."
        },
        "destinationBranch": {
          "type": "string",
          "description": "The id of destination branch of this PR."
        },
        "reviewers": {
          "type": "array",
          "items": {
            "$ref": "#/definitions/Reviewer"
          }
        },
        "lastUpdate": {
          "type": "string",
          "description": "The last time this PR was updated, used to
display the date of the latest activity for this PR. Formatted as a UTC
ISO 8601 date time format."
        },
        "timestamp": {
          "type": "string",
          "description": "A timestamp for this entity in ISO 8601 date
time format. This is used to drop out of order requests on a per entity
basis"
        },
        "id": {
          "type": "string",
          "description": "An id that is defined for this entity. This id
will be used for any cross entity linking"
        }
      }
    },
    "DevInformation": {
      "type": "object",
      "properties": {
        "commits": {
          "type": "array",
          "description": "The commits for this request",
          "items": {
```

```
            "$ref": "#/definitions/Commit"
          }
        },
        "repositories": {
          "type": "array",
          "description": "Information about the repositories referenced
by the dev information in this request",
          "items": {
            "$ref": "#/definitions/Repository"
          }
        },
        "branches": {
          "type": "array",
          "description": "The branches for this request",
          "items": {
            "$ref": "#/definitions/Branch"
          }
        },
        "preventTransitions": {
          "type": "boolean",
          "description": "prevent automatic transitions and smart
commits being fired for this request",
          "default": false
        },
        "pullRequests": {
          "type": "array",
          "description": "The pull requests for this request",
          "items": {
            "$ref": "#/definitions/PullRequest"
          }
        }
      }
    },
    "Branch": {
      "type": "object",
      "required": [
        "id",
        "lastCommit",
        "name",
        "repositoryId",
        "url"
      ],
      "properties": {
        "name": {
          "type": "string"
        },
        "repositoryId": {
          "type": "string",
          "description": "The id of the repository for this branch, will
be used to lookup the repository details when displaying this branch"
        },
        "url": {
          "type": "string",
```

```
          "description": "The url of the branch."
        },
        "createPullRequestUrl": {
          "type": "string",
          "description": "A url that is displayed to the user which will
take them to a page where they can create a pull request from this
branch"
        },
        "lastCommit": {
          "description": "The last commit on this branch, used to
display the date of the latest activity for this branch",
          "$ref": "#/definitions/Commit"
        },
        "timestamp": {
          "type": "string",
          "description": "A timestamp for this entity in ISO 8601 date
time format. This is used to drop out of order requests on a per entity
basis"
        },
        "id": {
          "type": "string",
          "description": "An id that is defined for this entity. This id
will be used for any cross entity linking"
        }
      }
    },
    "Reviewer": {
      "type": "object",
      "required": [
        "name"
      ],
      "properties": {
        "approved": {
          "type": "boolean",
          "description": "Whether this reviewer has approved the pull
request",
          "default": false
        },
        "name": {
          "type": "string"
        },
        "avatar": {
          "type": "string",
          "description": "A url to the avatar for this user"
        },
        "url": {
          "type": "string",
          "description": "A url to the profile for this user"
        }
      }
    },
    "Repository": {
      "type": "object",
```

```
      "required": [
        "name",
        "url"
      ],
      "properties": {
        "name": {
          "type": "string"
        },
        "id": {
          "type": "string",
          "description": "An id that identifies this repository. This id
will be used for in other  entities that need to reference the
repository"
        },
        "avatar": {
          "type": "string",
          "description": "A URL to the avatar for this repository"
        },
        "url": {
          "type": "string",
          "description": "A url to this repository for display"
        },
        "description": {
          "type": "string"
        },
        "forkOf": {
          "description": "The repository this repository was forked
from, if this repository is a fork",
          "$ref": "#/definitions/Repository"
        }
      }
    },
    "Commit": {
      "type": "object",
      "required": [
        "author",
        "authorTimestamp",
        "displayId",
        "fileCount",
        "hash",
        "id",
        "repositoryId",
        "url"
      ],
      "properties": {
        "hash": {
          "type": "string",
          "description": "The hash of the commit."
        },
        "flags": {
          "type": "array",
          "description": "The set of flags for this commit that supply
any additional context",
```

```
          "uniqueItems": true,
          "items": {
            "type": "string",
            "enum": [
              "MERGE_COMMIT"
            ]
          }
        },
        "message": {
          "type": "string"
        },
        "repositoryId": {
          "type": "string",
          "description": "The id of the repository for this commit, will
be used to lookup the repository details when displaying this commit"
        },
        "author": {
          "description": "Details of the author of this commit",
          "$ref": "#/definitions/Author"
        },
        "authorTimestamp": {
          "type": "string",
          "description": "The Author date of this commit. Formatted as a
UTC ISO 8601 date time format."
        },
        "displayId": {
          "type": "string",
          "description": "A shortened identifier for the commit, used
for display"
        },
        "url": {
          "type": "string",
          "description": "The url for this commit"
        },
        "files": {
          "type": "array",
          "items": {
            "$ref": "#/definitions/File"
          }
        },
        "fileCount": {
          "type": "integer",
          "format": "int32",
          "description": "The total number of files added, removed or
modified by this commit"
        },
        "timestamp": {
          "type": "string",
          "description": "A timestamp for this entity in ISO 8601 date
time format. This is used to drop out of order requests on a per entity
basis"
        },
        "id": {
```

```
          "type": "string",
          "description": "An id that is defined for this entity. This id
will be used for any cross entity linking"
        }
      }
    },
    "Author": {
      "type": "object",
      "required": [
        "name"
      ],
      "properties": {
        "name": {
          "type": "string"
        },
        "avatar": {
          "type": "string",
          "description": "A url to the avatar for this user"
        },
        "url": {
          "type": "string",
          "description": "A url to the profile for this user"
        }
      }
    },
    "File": {
      "type": "object",
      "properties": {
        "path": {
          "type": "string",
          "description": "The file path that is displayed to the user"
        },
        "url": {
          "type": "string",
          "description": "A url for this file"
        },
        "changeType": {
          "type": "string",
          "enum": [
            "ADDED",
            "COPIED",
            "DELETED",
            "MODIFIED",
            "MOVED",
            "UNKNOWN"
          ]
        },
        "linesAdded": {
          "type": "integer",
          "format": "int32"
        },
        "linesRemoved": {
          "type": "integer",
```

```
            "format": "int32"
          }
        }
```

```
      }
    }
  }
```

The main API that should be used is `/external/devInformation/{issueKey}` which can be used to supply Commits, Branches and Pull Requests to JIRA. There is some cross referencing between the entities, particularly to match information for repositories against Commits which will be done through the `id` attribute.

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`.

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 Unknown	Pull request reopened.
Pull Request in request with status merged and existing pull request with status open	Pull request merged.
Pull Request in request with status declined and existing pull request with status open	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	
Branch in request		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.

Sample request.

This request shows a sample json payload to create a commit, branch and pull request against an issue in JIRA.

Click here to expand...


```
{
  "preventTransitions" : false,
  "repositories" : [ {
    "id" : "https://bitbucket.org/benjaminmorgan/foo",
    "url" : "https://bitbucket.org/benjaminmorgan/foo",
    "name" : "foo",
    "avatar" :
"https://i0.wp.com/avatar-cdn.atlassian.com/default/32?ssl=1",
    "description" : "The foo repository"
  } ],
  "commits" : [ {
    "url" :
"https://api.bitbucket.org/2.0/repositories/benjaminmorgan/foo/commit/62
c52d83ca58b53a5bd441164e2390f3788bcbd7",
    "repositoryId" : "https://bitbucket.org/benjaminmorgan/foo",
    "hash" : "62c52d83ca58b53a5bd441164e2390f3788bcbd7",
    "displayId" : "62c52d",
    "authorTimestamp" : "2017-10-12T02:56:51+00:00",
    "author" : {
      "name" : "Benjamin Morgan",
      "avatar" :
"https://secure.gravatar.com/avatar/5366c2b021a0731dadb2ec7e84487ab1?r=g
&d=https://avatar-cdn.atlassian.com/default/32&s=32",
      "url" : "https://bitbucket.org/benjaminmorgan/"
    },
    "fileCount" : 1,
    "flags" : [ ],
    "message" : "FCC-1 some change",
    "files" : [ {
      "path" : "/something",
      "url" :
"https://bitbucket.org/benjaminmorgan/foo/commits/30d959a5ba0c2109875980
5cff6904ff3c194a28?at=wibble#chg-something",
      "changeType" : "ADDED",
      "linesAdded" : 1,
      "linesRemoved" : 0
    } ],
    "id" :
"https://api.bitbucket.org/2.0/repositories/benjaminmorgan/foo/commit/62
c52d83ca58b53a5bd441164e2390f3788bcbd7",
    "timestamp" : "2017-10-12T02:56:51+00:00"
  } ],
  "branches" : [ {
    "url" :
```

```
"https://api.bitbucket.org/2.0/repositories/benjaminmorgan/foo/refs/bran
ches/wibble",
    "repositoryId" : "https://bitbucket.org/benjaminmorgan/foo",
    "name" : "wibble",
    "createPullRequestUrl" :
"https://bitbucket.org/benjaminmorgan/foo/pull-requests/new?source=benja
minmorgan/foo%3A%3Awibble&dest=benjaminmorgan/foo%3A%3Amaster",
    "lastCommit" : {
      "url" :
"https://api.bitbucket.org/2.0/repositories/benjaminmorgan/foo/commit/62
c52d83ca58b53a5bd441164e2390f3788bcbd7",
      "repositoryId" : "https://bitbucket.org/benjaminmorgan/foo",
      "hash" : "62c52d83ca58b53a5bd441164e2390f3788bcbd7",
      "displayId" : "62c52d",
      "authorTimestamp" : "2017-10-12T02:56:51+00:00",
      "author" : {
        "name" : "Benjamin Morgan",
        "avatar" :
"https://secure.gravatar.com/avatar/5366c2b021a0731dadb2ec7e84487ab1?r=g
&d=https://avatar-cdn.atlassian.com/default/32&s=32",
        "url" : "https://bitbucket.org/benjaminmorgan/"
      },
      "fileCount" : 1,
      "flags" : [ ],
      "message" : "FCC-1 some change",
      "files" : [ {
        "path" : "/something",
        "url" :
"https://bitbucket.org/benjaminmorgan/foo/commits/30d959a5ba0c2109875980
5cff6904ff3c194a28?at=wibble#chg-something",
        "changeType" : "ADDED",
        "linesAdded" : 1,
        "linesRemoved" : 0
      } ],
      "id" :
"https://api.bitbucket.org/2.0/repositories/benjaminmorgan/foo/commit/62
c52d83ca58b53a5bd441164e2390f3788bcbd7",
      "timestamp" : "2017-10-12T02:56:51+00:00"
    },
    "id" :
"https://api.bitbucket.org/2.0/repositories/benjaminmorgan/foo/refs/bran
ches/wibble"
  } ],
  "pullRequests" : [ {
    "url" :
"https://api.bitbucket.org/2.0/repositories/benjaminmorgan/foo/pullreque
sts/1",
    "repositoryId" : "https://bitbucket.org/benjaminmorgan/foo",
    "displayId" : "1",
    "author" : {
      "name" : "Benjamin Morgan",
      "avatar" :
"https://secure.gravatar.com/avatar/5366c2b021a0731dadb2ec7e84487ab1?r=g
```

```
&d=https://avatar-cdn.atlassian.com/default/32&s=32",
      "url" : "https://bitbucket.org/benjaminmorgan/"
    },
    "title" : "FCC-1 some change",
    "commentCount" : 1,
    "sourceBranch" :
"https://api.bitbucket.org/2.0/repositories/benjaminmorgan/foo/refs/bran
ches/wibble",
    "destinationBranch" :
"https://api.bitbucket.org/2.0/repositories/benjaminmorgan/foo/refs/bran
ches/master",
    "reviewers" : [ {
      "name" : "Bob",
      "avatar" :
"https://i0.wp.com/avatar-cdn.atlassian.com/default/32?ssl=1",
      "approved" : false
    } ],
    "status" : "OPEN",
    "lastUpdate" : "2017-10-12T02:56:51+00:00",
    "id" :
"https://api.bitbucket.org/2.0/repositories/benjaminmorgan/foo/pullreque
```

```
  sts/1"
    } ]
  }
```