

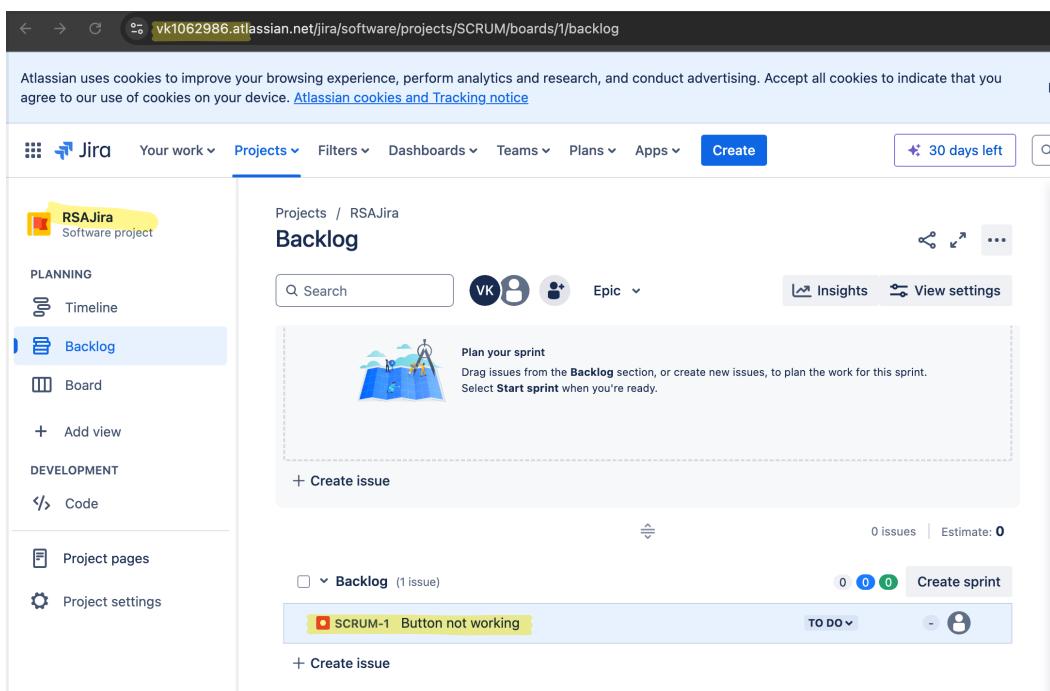
Rest Assured API advanced

Jira

- ✓ How to send File Attachments through Rest API calls
- ✓ Example used: Jira API to create Bug and attached failed screenshots through Rest APIs

<https://vk1062986.atlassian.net/jira/software/projects/SCRUM/boards/1/backlog?selectedIssue=SCRUM-1>

<https://developer.atlassian.com/cloud/jira/platform/rest/v2/intro/#about>



>Basic Auth & OAuth authentication are 2 types used to unlock APIs.

>We use Basic Auth to unlock Jira APIs

<https://developer.atlassian.com/cloud/jira/platform/basic-auth-for-rest-apis/>

<https://id.atlassian.com/manage-profile/security/api-tokens>

>Basically when we say Basic Authentication that means we will be able to login using Tokens.

Here we usually call tokens as Bearer tokens

<https://id.atlassian.com/manage-profile/security/api-tokens>

Secret key for API token created in Jira:

ATATT3xFfGF0rvYTdYHyuHwizA8wWuHHUZLqoZq-
IhdZ39Xw6gUFQPnM6SR4XntU8KLLuAlvQ-
F9nsEnZI4YZT2pYCS92Ga0dSf1FyU5EuFhPvYChFp2QIFMxDByKVHxQoQxyi
FxXI1CkpDdhkGqHUmZvS1kE0gzvSjfiH0sSjAefgw5f3Yhcbo=3602714E

Supply basic auth headers in Jira

<https://developer.atlassian.com/cloud/jira/platform/basic-auth-for-rest-apis/>

The screenshot shows a web browser window with the URL <https://developer.atlassian.com/cloud/jira/platform/basic-auth-for-rest-apis/>. The page content is as follows:

Supply basic auth headers

You can construct and send basic auth headers. To do this you perform the following steps:

1. Generate an API token for Jira using your [Atlassian Account](#).
2. Build a string of the form `useremail:api_token`.
3. BASE64 encode the string.
 - Linux/Unix/MacOS:

```
1 echo -n user@example.com:api_token_string | base64
```
 - Windows 7 and later, using Microsoft Powershell:

```
1 $Text = 'user@example.com:api_token_string'
2 $Bytes = [System.Text.Encoding]::UTF8.GetBytes($Text)
3 $EncodedText = [Convert]::ToBase64String($Bytes)
4 $EncodedText
```
4. Supply an Authorization header with content Basic followed by the encoded string. For example, the string fred:fred encodes to ZnJlZDpmcmVk in base64, so you would make the request as follows:

```
curl -D- \
-X GET \
-H "Authorization: Basic ZnJlZDpmcmVk" \
-H "Content-Type: application/json" \
"https://your-domain.atlassian.net/rest/api/2/issue/0A-31"
```

ON THIS PAGE

- Overview
- Get an API token
- Simple example
- Supply basic auth headers**
- Advanced topics

OTHER CONSIDERATIONS

- Atlassian Design Guidelines
- Atlaskit
- Data residency
- Atlassian Marketplace
- Cloud app licensing
- Developer canary program
- Developing apps for Jira Cloud mobile

Advanced topics

[Authentication challenges](#)

You can construct and send basic auth headers. To do this you perform the following steps:

1. Generate an API token for Jira using your [Atlassian Account](#).
2. Build a string of the form `useremail:api_token`.
3. BASE64 encode the string.
 - Linux/Unix/MacOS:

`echo -n user@example.com:api_token_string | base64`

- Windows 7 and later, using Microsoft Powershell:

```
$Text = 'user@example.com:api_token_string'
$Bytes = [System.Text.Encoding]::UTF8.GetBytes($Text)
$EncodedText = [Convert]::ToBase64String($Bytes)
$EncodedText
```

4. Supply an Authorization header with content Basic followed by the encoded string. For example, the string fred:fred encodes to ZnJlZDpmcmVk in base64, so you would make the request as follows:

```
curl -D- \
-X GET \
-H "Authorization: Basic ZnJlZDpmcmVk" \
-H "Content-Type: application/json" \
```

```
"https://your-domain.atlassian.net/rest/api/2/issue/  
QA-31"
```

Actual hands on

- Syntax:

```
useremail:api_token
```

- Actual token:

```
vk1062986@gmail.com:ATATT3xFfGF0rvYTdYHyuHwizA8wWuHHUZLqoZq-  
IhdZ39Xw6gUFQPnM6SR4XntU8KLLuAlvQ-  
F9nsEnZI4YZT2pYCS92Ga0dSf1FyU5EuFhPvYChFp2QIFMxDByKVHxQoQxyi  
FxXI1CkpDdhkGqHUmZvS1kE0gzvSjfIH0sSjAefgw5f3Yhcbo=3602714E
```

- Syntax:

```
echo -n user@example.com:api_token_string | base64
```

- Actual command

```
echo -n
```

```
vk1062986@gmail.com:ATATT3xFfGF0rvYTdYHyuHwizA8wWuHHUZLqoZq-  
IhdZ39Xw6gUFQPnM6SR4XntU8KLLuAlvQ-  
F9nsEnZI4YZT2pYCS92Ga0dSf1FyU5EuFhPvYChFp2QIFMxDByKVHxQoQxyi  
FxXI1CkpDdhkGqHUmZvS1kE0gzvSjfIH0sSjAefgw5f3Yhcbo=3602714E |  
base64
```

<https://www.base64decode.org/>

>We can use online base64 encode & decoder to create base64 encoded format or execute above command in terminal to get base64 encoded format

base64encode.org

Encode to Base64 format

Simply enter your data then push the encode button.

```
vk1062986@gmail.com:ATATT3xFFGF0rvYTdYHyuHwizA8wWuHHUZLqoZq-IhdZ39Xw6gUFQPnM6SR4XntU8KLLuAlvQ-F9nsEnZl4YZT2pYCS92Ga0dSf1FyU5EuFhpVYChfp2QIFMxDByKVHxQoQxyiFxIICkpDdhkGqHUmZvS1kEOgvSjflHosSjAefgw5f3Yhbo=3602714E
```

To encode binaries (like images, documents, etc.) use the file upload form a little further down on this page.

UTF-8 Destination character set.

LF (Unix) Destination newline separator.

Encode each line separately (useful for when you have multiple entries).

Split lines into 76 character wide chunks (useful for MIME).

Perform URL-safe encoding (uses Base64URL format).

Live mode OFF Encodes in real-time as you type or paste (supports only the UTF-8 character set).

ENCODE Encodes your data into the area below.

```
dmsxMDYyOTg2QGdtYWlsLmNvbTpBVEFUDN4RmZHRjByd1lUZF1IeXVId216QTh3V3VISFVaTHFvWnEtSWhkWjM5Whc2Z1VGUVBuTTZTUjRYbnRVOEtMTHVBbHZRLUY5bnNFb1pJNFlaVDJwWUNTOTJHYTBkU2YxRnlVNUV1RmhQd11DaEZwM1FJRK14REJ5S1Z1eFFvUXh5aUZ4WElsQ2twRGRo0dxSFVtWnZTMwtFT2d6d1NqZk1IMHNTakFlZmd3NWYzWWhjYm89MzYwMjcxNEU=
```

Converted base64 Encoded format from string format of Api token for MAC:

```
krishnabros@Krishnas-MacBook-Air RestAssured_DemoProject % cd ~
krishnabros@Krishnas-MacBook-Air ~ % echo -n vk1062986@gmail.com:ATATT3xFFGF0rvYTdYHyuHwizA8wWuHHUZLqoZq-IhdZ39Xw6gUFQPnM6SR4XntU8KLLuAlvQ-F9nsEnZl4YZT2pYCS92Ga0dSf1FyU5EuFhpVYChfp2QIFMxDByKVHxQoQxyiFxIICkpDdhkGqHUmZvS1kEOgvSjflHosSjAefgw5f3Yhbo=3602714E | base64
dmsxMDYyOTg2QGdtYWlsLmNvbTpBVEFUDN4RmZHRjByd1lUZF1IeXVId216QTh3V3VISFVaTHFvWnEtSWhkWjM5Whc2Z1VGUVBuTTZTUjRYbnRVOEtMTHVBbHZRLUY5bnNFb1pJNFlaVDJwWUNTOTJHYTBkU2YxRnlVNUV1RmhQd11DaEZwM1FJRK14REJ5S1Z1eFFvUXh5aUZ4WElsQ2twRGRo0dxSFVtWnZTMwtFT2d6d1NqZk1IMHNTakFlZmd3NWYzWWhjYm89MzYwMjcxNEU=
```

Converted Base64 Encoded format

```
dmsxMDYyOTg2QGdtYWlsLmNvbTpBVEFUDN4RmZHRjByd1lUZF1IeXVId216QTh3V3VISFVaTHFvWnEtSWhkWjM5Whc2Z1VGUVBuTTZTUjRYbnRVOEtMTHVBbHZRLUY5bnNFb1pJNFlaVDJwWUNTOTJHYTBkU2YxRnlVNUV1RmhQd11DaEZwM1FJRK14REJ5S1Z1eFFvUXh5aUZ4WElsQ2twRGRo0dxSFVtWnZTMwtFT2d6d1NqZk1IMHNTakFlZmd3NWYzWWhjYm89MzYwMjcxNEU=
```

> Above is header we need to pass to Authentication header value of API.

Go to Issues documentation from below link:

<https://developer.atlassian.com/cloud/jira/platform/rest/v3/api-group-issues/#api-rest-api-3-events-get>

Test:

✓ In postman create issue post request: **POST**

<https://developer.atlassian.com/cloud/jira/platform/rest/v3/api-group-issues#api-rest-api-3-events-get>

REST API v3 (beta)

- Issue resolutions
- Issue search
- Issue security level
- Issue security schemes
- Issue type properties
- Issue type schemes
- Issue type screen schemes
- Issue types
- Issue votes
- Issue watchers
- Issue working properties
- Issue worklogs
- Issues**
 - GET** Get events
 - POST** Create issue
 - PUT** Archive issue(s) by issue ID/key
 - POST** Archive issue(s) by JQL
 - POST** Bulk create issue
 - POST** Bulk fetch issues
 - GET** Get create issue metadata
 - GET** Get create metadata issue types for a project
 - GET** Get create field metadata for a project

401 Unauthorized

403 Forbidden

Create issue

Creates an issue or, where the option to create subtasks is enabled in Jira, a subtask. A transition may be applied, to move the issue or subtask to a workflow step other than the default start step, and issue properties set.

The content of the issue or subtask is defined using `update` and `fields`. The fields that can be set in the issue or subtask are determined based on the `Get create issue` endpoint, which are the same as those available on the issue's basic screen. Note that the `description`, `environment`, and any `textarea` custom fields (multi-line text fields) take Atlassian Document Format content. Single line custom fields (`textfield`) accept a string and don't handle Atlassian Document Format content.

Creating a subtask differs from creating an issue as follows:

- `issuetype` must be set to a subtask issue type (use `Get create issue` metadata to find subtask issue types)
- parent must contain the ID or key of the parent issue.

In a next-gen project any issue may be made a child providing that the parent and child are members of the same project.

Permissions required: `Browse projects` and `Create issues` project permissions for the project in which the issue or subtask is created.

Data Security Policy: Not exempt from app access rules

Scopes:

- Connect app scope required: `WRITE`
- OAuth 2.0 scopes required:
- Classic **RECOMMENDED**: `write:jira-work`

POST /rest/api/3/issue

Forge curl Node.js Java Python PHP

```
curl --request POST \
--url https://your-domain.atlassian.net/rest/api/3/issue \
--user 'email@example.com:api_token' \
--header 'Accept: application/json' \
--header 'Content-Type: application/json' \
--data '{
  "fields": {
    "assignee": {
      "id": "5b189f2e9729b51b54dc274d"
    },
    "components": [
      {
        "id": "10000"
      }
    ],
    "customfield_10000": "09/Jun/19",
    "customfield_20000": "06/Jul/19 3:26 PM",
    "customfield_30000": {
      "id": "10000",
      "value": "10000"
    },
    "customfield_40000": {
      "content": [
        {
          "text": "Occurs on all orders",
          "type": "text"
        }
      ]
    }
  }
}'
```

<https://developer.atlassian.com/server/jira/platform/jira-rest-api-example-create-issue-7897248/>

Can ask for, by specifying the project key, project names, issue type keys, or issue type names in the URL, for example, to get the create metadata for the Bug issue type in the JRA project:

```
1 http://localhost:8090/rest/api/2/issue/createmeta?projectKeys=JRA&issueTypeNames=Bug&expand=prc
```

See the [Discovering meta-data for creating issues](#) tutorial for more information

Examples of creating an issue

Example of creating an issue using project keys and field names.

This simple create request

Request

```
1 curl -D- -u fred:fred -X POST --data (see below) -H "Content-Type: application/json" http://loc
```

Data

```
1 {
  "fields": {
    "project": {
      "key": "TEST"
    },
    "summary": "REST ye merry gentlemen.",
    "description": "Creating of an issue using project keys and issue type names using the REST API",
    "issuetype": {
      "name": "Bug"
    }
}
```

Response

```
1 {
  "id": "390000",
  "key": "TEST-01",
  "self": "https://localhost:8090/rest/api/2/issue/390000"
```

<https://vk1062986.atlassian.net/jira/software/projects/SCRUM/settings/details>

Atlassian uses cookies to improve your browsing experience, perform analytics and research, and conduct advertising. Accept all cookies to indicate that you agree to our use of cookies on your device. [Atlassian cookies and Tracking notice](#)

Preferences Only necessary

Jira Your work ▾ Projects ▾ Filters ▾ Dashboards ▾ Teams ▾ Plans ▾ Apps ▾ Create 30 days left Search

RSAJira Software project

Back to project

Details

Required fields are marked with an asterisk *

Name * RSAJira

Key * SCRUM

Category Choose a category

The screenshot shows the Jira Backlog interface for the project 'RSAJira'. On the left, the sidebar includes sections for Planning, Backlog, Development, and Project pages. The main area displays the backlog with a single issue titled 'Button not working' under the 'Backlog' section. The issue has a status of 'TO DO' and is assigned to 'SCRM-1'. A yellow highlight is placed over the 'TO DO' status indicator.

The screenshot shows the Postman API client interface. A POST request is being made to 'https://vk1062986.atlassian.net/rest/api/3/issue' with the 'Create Bug' endpoint selected. The request body contains JSON data for creating a new issue. The response shows a '201 Created' status with a response time of 694 ms and a response size of 1011 B. The response body shows the newly created issue with id '10001', key 'SCRM-2', and self-link 'https://vk1062986.atlassian.net/rest/api/3/issue/10001'. To the right, a code snippet panel displays the full cURL command used for the request.

The screenshot shows the Jira Backlog interface for the project 'RSAJira'. The backlog now contains two issues: 'Button not working' and 'Dropdowns are not working', both under the 'Backlog' section. Both issues have a status of 'TO DO' and are assigned to 'SCRM-1'. A yellow highlight is placed over the 'TO DO' status indicator for the second issue.

Below is the cURL command to Create Issue or Bug in Jira:

```
curl --location 'https://vk1062986.atlassian.net/rest/api/3/issue' \
```

```

--header 'Content-Type: application/json' \
--header 'Authorization: Basic
dmsxMDYyOTg2QGdtYWlsLmNvbTpBVEFUVDN4RmZHRjByd1lUZF1IeXVIId216
QTh3V3VISFVaTHFvWnEtSWhkjM5WHc2Z1VGUVBuTTZTUjRYbnRVOEtMTHVB
bHZRLUY5bnNFb1pJNFlaVDJwwUNTOtJHYTBkU2YxRn1VNUV1RmhQd11DaEZw
M1FJRk14REJ5S1ZIeFFvUXh5aUZ4WElsQ2twRGRo0dxSFVtWnZTMwtFT2d6
d1NqZk1IMHNTakF1Zmd3NWYzWWhjYm89MzYwMjcxNEU=' \
--header 'Cookie:
atlassian.xsrf.token=b67d6a51ab6eb0501bbdd753f1154966f43b0cd
d_lin' \
--data '{
    "fields": {
        "project": {
            "key": "SCRUM"
        },
        "summary": "Dropdowns are not working",
        "issuetype": {
            "name": "Bug"
        }
    }
}
'

```

✓ In postman, get the issue : **GET**

```

curl --request GET \
--url 'https://your-domain.atlassian.net/rest/api/3/issue/{issueIdOrKey}' \
--user 'email@example.com:<api_token>' \
--header 'Accept: application/json'

```

The screenshot shows the Postman interface with the following details:

- Left Sidebar:** REST API v3 (beta) > Issues > Get issue.
- Request URL:** `/rest/api/3/issue/{issueIdOrKey}`
- Request Headers:** Accept: application/json
- Code Block:**

```

1 curl --request GET \
2 --url 'https://your-domain.atlassian.net/rest/api/3/issue/{issueIdOrKey}' \
3 --user 'email@example.com:<api_token>' \
4 --header 'Accept: application/json'

```
- Response Preview:** Shows a JSON response structure for an issue, including fields like id, type, title, and labels.

Below is the cURL command to Get issue using id from Jira:

```

curl --location 'https://vk1062986.atlassian.net/rest/api/3/
issue/10001' \

```

```
--header 'Authorization: Basic  
dmsxMDYyOTg2QGdtYWlsLmNvbTpBVEFUVDN4RmZHRjBydl1UZF1IeXVIId216  
QTh3V3VISFVaTHFvWnEtSWhkWjM5WHc2Z1VGUVButTzTUjRYbnRVOEtMTHVB  
bHZRLUY5bnNFb1pJNF1aVDJwWUNTOTJHYTBkU2YxRn1VNUV1RmhQd11DaEZw  
M1FJRK14REJ5S1ZIeFFvUXh5aUZ4WE1sQ2twRGRoA0dxSFVtWnZTMWtFT2d6  
d1NqZk1IMHNTakF1Zmd3NWYzWWhjYm89MzYwMjcxEtNEU=' \  
--header 'Cookie:  
atlassian.xsrf.token=6ba5b563952d94a8d5d020c76163a736df4f25a  
3_lin'
```

Below is the cURL command to Get issue using key from Jira:

```
curl --location 'https://vk1062986.atlassian.net/rest/api/3/issue/SCRUM-1' \
--header 'Authorization: Basic dmsxMDYyOTg2QGdtYWlsLmNvbTpBVEFUVDN4RmZHRjByd1lUZF1IeXVIId216QTh3V3VISFVaTHFvWnEtSWhkWjM5WHc2Z1VGUVBuTTZTUjRYbnRVOEtMTHVBbHZRLUY5bnNFb1pJNF1aVDJwWUNTOTJHYTBkU2YxRn1VNUV1RmhQd1lDaEZwM1FJRK14REJ5S1ZIeFFvUXh5aUZ4WElsQ2twRGRoA0dxSFVtWnZTMWtFT2d6d1NqZk1IMHNTakF1Zmd3NWYzWWhjYm89MzYwMjcxNEU=' \
--header 'Cookie: atlassian.xsrf.token=6ba5b563952d94a8d5d020c76163a736df4f25a3_lin'
```

The screenshot shows the Postman application interface. At the top, there's a navigation bar with 'Home', 'Workspaces', 'API Network', and a search bar 'Search Postman'. On the far right are 'Invite', 'Settings', 'Logout', and 'Upgrade' buttons.

The main workspace is titled 'My Workspace'. It contains several collections and environments:

- Collections: 'Library API created by me', 'RSAMaps', 'Contract Testing', 'Integration Testing', 'Intro to Writing Tests', 'Jira API', and 'POST Create Bug'.
- Environments: 'Collections', 'History', and 'APIs'.

The 'POST Create Bug' collection is currently selected. Under it, there are two items: 'Create Bug' and 'Get Bug'. The 'Get Bug' item is highlighted with a yellow background and has a dropdown arrow icon next to it.

In the center, a request card for 'Jira API / Get Bug' is displayed. The method is 'GET', the URL is 'https://vk1062986.atlassian.net/rest/api/3/issue/10001', and the status is '200 OK' with a response time of '381 ms'. Below the URL, there are tabs for 'Params', 'Auth', 'Headers (9)', 'Body', 'Scripts', 'Settings', and 'Cookies'. The 'Body' tab is selected, showing a JSON response with a summary field containing the value 'Dropdowns are not working.'.

To the right of the request card, a 'Code snippet' panel is open, showing a curl command to make the same API call. The command includes headers for Authorization and a cookie for 'atlassian.xsrf'. The curl command is as follows:

```
curl --location 'https://vk1062986.atlassian.net/rest/api/3/issue/10001' \
--header 'Authorization: Basic dnxMDY0Tg0GtW1LmWvbTpBVEFUVDN4RnzHRjbYd1lUf2IeXtId2l0qTh3V3ISFVaTHFvhnEtSwkhej5Ww-221WVBU0rYRvNvOEHTMhvBnB2HRLy5mNbpJ3mIvDwMNUT0jHTBkU2YRn1WVUVNrmQd1l0EzWf1MKRk4R3S1zL1gPFVJKh6u24We1Q2tWvRGoadeKsPvTwhn7mtrfTfdded1mQ-11MWHMaF1Znd3NWYzWhnYm89MzYwjqcNEu-`
```

Below the curl command, there's a note: '3 -> header 'Cookie: atlassian.xsrf. token=ea50563952d94a0d5020c76163a736d4425a_1_in'

The screenshot shows the Postman interface with a successful API call to the Jira API. The URL is `https://vk1062986.atlassian.net/rest/api/3/issue/SCRM-1`. The response status is 200 OK, with a response time of 779 ms and a size of 2.29 KB. The response body is a JSON object representing a bug with various fields like summary, creator, and attachments.

```

{
  "id": "103",
  "key": "SCRM-1",
  "type": "bug",
  "status": "new",
  "priority": "minor",
  "summary": "Button not working",
  "description": "The button is not responding to click events. It appears functional but does not trigger any action.", "customfield_10004": null,
  "customfield_10005": null,
  "customfield_10006": null,
  "customfield_10007": null,
  "customfield_10008": null,
  "customfield_10009": null,
  "aggregatetimeestimate": null,
  "attachment": [],
  "summary": "Button not working",
  "creator": {
    "self": "https://vk1062986.atlassian.net/rest/api/3/user/account",
    "accountId": "12026711e2c2-0d00-47c0-a0ff-db56f2b97d6e",
    "emailAddress": "vk1062986@gmail.com",
    "avatarUrls": {
      "48x48": "https://secure.gravatar.com/avatar/04edbbc3fae2339464a48",
      "24x24": "https://secure.gravatar.com/avatar/04edbbc3fae2339464a48",
      "16x16": "https://secure.gravatar.com/avatar/04edbbc3fae2339464a48",
      "32x32": "https://secure.gravatar.com/avatar/04edbbc3fae2339464a48"
    },
    "displayName": "Vinay Krishna",
    "active": true,
    "timeZone": "Asia/Calcutta"
  }
}

```

✓ In postman, Add Screenshot : POST

```
curl --request POST \
--url 'https://your-domain.atlassian.net/rest/api/3/issue/{issueIdOrKey}/attachments' \
--user 'email@example.com:<api_token>' \
--header 'Accept: application/json'
```

```
curl --location --request POST 'https://your-domain.atlassian.net/rest/api/3/issue/TEST-123/attachments'
-u 'email@example.com:<api_token>'
-H 'X-Atlassian-Token: no-check'
--form 'file=@"myfile.txt"'
```

The screenshot shows the Jira REST API v3 documentation for the `/rest/api/3/issue/{issueIdOrKey}/attachments` endpoint. It provides a curl example and a Node.js script for uploading attachments to an issue.

```

curl --request POST \
--url 'https://your-domain.atlassian.net/rest/api/3/issue/TEST-123/attachments' \
-u 'email@example.com:<api_token>' \
-H 'X-Atlassian-Token: no-check'
--form 'file=@"myfile.txt"'

```

```

// This code sample uses the 'node-fetch' and 'FormData' library
// https://www.npmjs.com/package/node-fetch
// https://www.npmjs.com/package/form-data
const fetch = require('node-fetch');
const FormData = require('form-data');
const fs = require('fs');

const filePath = 'myfile.txt';
const form = new FormData();
const file = fs.createReadStream(filePath);
form.append('file', file);

```

The screenshot shows the Postman interface with a POST request to <https://vk1062986.atlassian.net/rest/api/3/issue/10001/attachments>. The 'Body' tab is selected with 'form-data' chosen. A file named 'Screenshot 2024-09-30 at 4.57.09 AM.png' is attached. The response shows a 200 OK status with a JSON payload containing issue details and the uploaded file's metadata.

```

[{"id": "10000", "filename": "Screenshot 2024-09-30 at 4.57.09 AM.png", "author": {"self": "https://vk1062986.atlassian.net/rest/api/3/user/710202083A71ee2c2-0d00-47c0-a80f-db5e2", "accountId": "710202083A71ee2c2-0d00-47c0-a80f-db5e2", "emailAddress": "vk1062986@gmail.com", "avatarUrls": {"32x32": "https://secure.gravatar.com/avatar/04edbccfa233975c584e178ae3103d2dchtips3A8C2F%2Favata", "24x24": "https://secure.gravatar.com/avatar/04edbccfa233975c584e178ae3103d2dchtips3A8C2F%2Favata", "16x16": "https://secure.gravatar.com/avatar/04edbccfa233975c584e178ae3103d2dchtips3A8C2F%2Favata", "8x8": "https://secure.gravatar.com/avatar/04edbccfa233975c584e178ae3103d2dchtips3A8C2F%2Favata"}, "displayName": "Vinay Krishna", "active": true, "timeZone": "Asia/Calcutta", "accountType": "atlassian"}, "self": "https://vk1062986.atlassian.net/rest/api/3/attachment/10000", "id": "10000", "filename": "Screenshot 2024-09-30 at 4.57.09 AM.png", "author": {"self": "https://vk1062986.atlassian.net/rest/api/3/user/710202083A71ee2c2-0d00-47c0-a80f-db5e2", "accountId": "710202083A71ee2c2-0d00-47c0-a80f-db5e2", "emailAddress": "vk1062986@gmail.com", "avatarUrls": {"32x32": "https://secure.gravatar.com/avatar/04edbccfa233975c584e178ae3103d2dchtips3A8C2F%2Favata", "24x24": "https://secure.gravatar.com/avatar/04edbccfa233975c584e178ae3103d2dchtips3A8C2F%2Favata", "16x16": "https://secure.gravatar.com/avatar/04edbccfa233975c584e178ae3103d2dchtips3A8C2F%2Favata", "8x8": "https://secure.gravatar.com/avatar/04edbccfa233975c584e178ae3103d2dchtips3A8C2F%2Favata"}, "displayName": "Vinay Krishna", "active": true, "timeZone": "Asia/Calcutta", "accountType": "atlassian"}]

```

The screenshot shows the Jira Backlog page for project RSAJira. An issue titled 'Dropdowns are not working' is visible in the backlog, which includes a screenshot of the affected interface.

Below is the cURL command to Add screenshot in Jira:

```

curl --location 'https://vk1062986.atlassian.net/rest/api/3/issue/10001/attachments' \
--header 'Authorization: Basic dmsxMDYyOTg2QGdtYWlsLmNvbTpBVEFUDN4RmZHRjByd1lUZF1IeXVId216QTh3V3VISFVaTHFvWnEtSWhkWjM5WHc2Z1VGUVBuTTZTUjRYbnRVOEtMTHVBbhZRLUY5bnNFblpJNFlaVDJwWUNTOTJHYTBkU2YxRn1VNUV1RmhQd1lDaEZwM1FJRK14REJ5S1ZIeFFvUXh5aUZ4WE1sQ2twRGRoa0dxSFVtWnZTMwtFT2d6dlNqZk1IMHNTakF1Zmd3NWYzWWhjYm89MzYwMjcxNEU=' \
--header 'X-Atlassian-Token: no-check' \
--header 'Cookie: atlassian.xsrf.token=6ba5b563952d94a8d5d020c76163a736df4f25a3_lin' \
--form 'file=@"/Users/krishnabros/Desktop/Screenshot 2024-09-30 at 4.57.09 AM.png"'

```

Interview Question:

1. When we want to send any file or any attachments through REST API, we can do it through form parameters
i.e use 'form-data' under request body section of Postman
2. Sending parameters using form parameters is always secure as this will not concatenate parameters in the request URL
3. Sending parameters using query parameters is not secure as this will concatenate parameters in the request URL

Note:

Choose the data type you need for your request body: **form data**, **URL-encoded**, **raw**, **binary**, or **GraphQL**.

<https://learning.postman.com/docs/sending-requests/create-requests/parameters/>

Form-data or multipart