3. Scenarios and advanced concepts

Check REST API availability

You can check the availability of the REST API by calling `/projects` without a token on your Polarion server in your browser:

```
https://[Your_Polarion_server]/polarion/rest/v1/projects/
```

- If the REST API is available, you receive a **401 Unauthorized** error.
- If the REST API is unavailable, you receive a **503 Service Unavailable** error.

  Along with a JSON body if Polarion is running without REST API or an Apache HTML page if all of Polarion is down. (With or without the REST API.)

Update using PATCH

You can update using PATCH via the resource URL. The resource URL is part of a request's body, and the resource's identification (**type** and **ID**) must match the resource described by the URL. The client can only update resources if it knows their URL and identification. (Typically by retrieving them using a `GET` request first.)

All fields are optional in `PATCH` requests. The client only sends the fields (attributes and relationships) you want to update.

Warning

Treat multi-valued fields carefully. See the **Update multi-valued fields** section below for more information about updating multi-valued fields.

Example

Update only the Severity of a Work Item:

- **Example Request:**

  `PATCH https://myhost/polarion/rest/v1/projects/MyProject/workitems/WI-555`
- **Request Body:**

```
{
  "data": {
    "type": "workitems",
    "id": "MyProject/WI-555",
    "attributes": {
      "severity": "critical"
    }
  }
}
```

Update multi-valued fields

To update a field, you completely replace an old value with a provided value. This is especially important for multi-valued fields that contain a list of values. When you replace multi-valued fields, the old list of values is entirely replaced by the new list you provide.

- If you want to add some values, you must provide a list containing all the old values and the new ones you want to add.

Example

Add mTest to the list of Work Item Assignees that already contains sDeveloper:

- **Example Request:**

  `PATCH https://myhost/polarion/rest/v1/projects/MyProject/workitems/WI-555`

- **Request Body:**

```
{
  "data": {
    "type": "workitems",
    "id": "MyProject/WI-555",
    "relationships": {
      "assignee": {
        "data": [
          //User already in the list of "assignee"
          {
            "id": "sDeveloper",
            "type": "users"
          },
          //New user to be added to the list of "assignee"
          {
            "id": "mTest",
            "type": "users"
          }
        ]
      }
    }
  }
}
```

- If you want to remove some values, you must provide a list containing all the old values minus those to be removed.

Example

Remove mTest from the list of Work Item Assignees that already contains sDeveloper and mTest:

- **Example Request:**

  `PATCH  https://myhost/polarion/rest/v1/projects/MyProject/workitems/WI-555`
- **Request Body:**

```
{
  "data": {
    "type": "workitems",
    "id": "MyProject/WI-555",
    "relationships": {
      "assignee": {
        "data": [
          //User that will remain on the list.
          {
            "id": "sDeveloper",
            "type": "users"
          }
        ]
      }
    }
  }
}
```

- If you want to completely empty the field, you can provide an empty list or null value to the field.

Example

Remove all users from the list of Work Item Assignees:

- **Example Request:**

  `PATCH https://myhost/polarion/rest/v1/projects/MyProject/workitems/WI-555`
- **Request Body:**

```
{
  "data": {
    "type": "workitems",
    "id": "MyProject/WI-555",
    "relationships": {
      "assignee": {
        "data": null
      }
    }
  }
}
```

Set empty field value

Most fields can be unset by using a `null` value described in the example above, but there are some exceptions.

Boolean fields do not support `null`. (Use `false` instead.)

Text field values do not support `null`. Instead, you can use `""` (empty) or set the whole field to `null`.

Relationships fields must contain a data member. In order to set an empty value, you can provide a `null` data member (`"data": null`).

An empty object or empty list is equivalent to `null`.

`"author": { "data": { } } == "author": { "data": null }` (Single valued relationships.)

`"categories": { "data": [ ] } == "categories": { "data": [ { } ] } == "categories": { "data": null }`

(Multi valued relationships.)

Example

Clear multiple fields of a PATCH Work Item

- **Example Request:**

  `PATCH https://myhost/polarion/rest/v1/projects/MyProject/workitems/WI-555`

- **Request Body:**

```
{
  "data": {
    "type": "workitems",
    "id": "MyProject/WI-555",
    "attributes": {
      "status": null, //normal field
      "description": null, //text field
      "textField": {
          "type": "text/plain",
          "value": ""
        },
      "booleanField": false, //boolean field
    },
    "relationships": {
        "categories": {
          "data": [ ] //multi-valued relationship field
        },
        "customRelationship": {
          "data": null //single-valued relationship field
        }
      }
    }
    }
  }
```

Delete resources

Delete a single resource

You can delete a single resource using `DELETE` via the resource URL. Deleting a single resource does not require any request body. Successful deletion of a resource results in an empty response with `204 - No Content` status code.

Example

Delete a linked Work Item:

- **Example Request:**

  `DELETE https://myhost/polarion/rest/v1/projects/myProject/workitems/WI-555/linkedworkitems/relates_to/myProject/WI-550`

- **Request Body:**

```
No request body
```

Delete multiple resources

You can delete multiple resources using `DELETE` via the resource collection URL. Deleting multiple resources at once requires a request body with the list of resource indentifications (Type and ID) you want to delete. Successful deletion of the resources results in an empty response with `204 - No Content` status code.

Example

Delete multiple linked Work Items:

- **Example Request:**

  `DELETE https://myhost/polarion/rest/v1/projects/myProject/workitems/WI-555/linkedworkitems`

- **Request Body:**

```
{
  "data": [
    {
      "type": "linkedworkitems",
      "id": "myProject/WI-555/relates_to/myProject/WI-550"
    },
    {
      "type": "linkedworkitems",
      "id": "myProject/WI-555/tests/myProject/WI-500"
    }
  ]
}
```

Linked Work Items

Links between **Work Items** are represented by separate resources: `linkedworkitems` (direct links), `backlinkedWorkItems`, `externallylinkedworkitems`, and `linkedoslcresources`. The **Work Item** has a relationship to these resources. To fetch them, you can use the `relationships` endpoint. There are also separate endpoints to manage directly linked **Work Items**, externally linked **Work Items**, and OSLC links contained within a **Work Item**.

Add Work Item links

Work Item links are created by creating `linkedworkitems` resources. You do this by sending a `POST` request to the `linkedworkitems` endpoint of the source **Work Item**. Multiple links can be created at once.

Example

Add one or more Work Item links:

- **Example Request:**

  `POST /polarion/rest/v1/projects/myProject/workitems/WI-1/linkedworkitems`

- **Request Body:**

```
{
    "data": [
      {
        "type": "linkedworkitems",
        "attributes": {
            "role": "relates_to"
        },
        "relationships": {
            "workItem": {
                "data": {
                    "type": "workitems",
                    "id": "myProject/WI-2"
                }
            }
        }
    },
        ...
    ]
}
```

Get target linked Work Items

To retrieve additional information about the links, or details about the linked **Work Items**, you must use the **Work Item** endpoint or the **linked Work Items** endpoint and a suitable combination of `include` and `field` query parameters.

Example

Get a Work Item, its links, and linked Work Items in a single request:

- **Example Request:**

  ```
  GET /polarion/rest/v1/projects/myProject/workitems/WI-1?
  include=linkedWorkItems,linkedWorkItems.workItem&fields[workitems]=title&fields[l
  ```

- **Response Body:**

```
{
    "links": {...},
    "data": {
        "type": "workitems",
        "id": "myProject/WI-1",
        "attributes": {
            "title": "My Work Item Title"
        },
        "links": {...}
    },
    "included": [
      {
        "type": "linkedworkitems",
        "id": "myProject/WI-1/relates_to/myProject/WI-2",
        "attributes": {
            "role": "relates_to"
        },
        "relationships": {
            "workItem": {
                "data": {
                    "type": "workitems",
                    "id": "myProject/WI-2"
                }
            }
        },
        "links": {...}
    },
    {
        "type": "workitems",
        "id": "myProject/WI-2",
        "attributes": {
```

```
                "title": "My Linked Work Item Title"
            },
            "links": {...}
        },
        ...
    ]
}
```

Example

Get Work Item links of a Work Item, with all target Work Item titles:

- **Example Request:**

  `GET /polarion/rest/v1/projects/myProject/workitems/WI-1/linkedworkitems?fields[linkedworkitems]=@all&fields[workitems]=title&include=workItem`

- **Response Body:**

```
{
    "links": {...},
    "data": [
      {
        "type": "linkedworkitems",
        "id": "myProject/WI-1/relates_to/myProject/WI-2",
        "attributes": {
            "role": "relates_to"
        },
        "relationships": {
            "workItem": {
                "data": {
                    "type": "workitems",
                    "id": "myProject/WI-2"
                }
            }
        },
        "links": {...}
    },
    ...],
    "included": [
      {
        "type": "workitems",
        "id": "myProject/WI-2",
        "attributes": {
            "title": "My Linked Work Item Title"
        },
        "links": {...}
    }
  ]
}
```

Patch Work Item link

You can patch revision and suspect attributes of a linked **Work Item**. See Update using PATCH for more information on how to perform patch operations.

Attachments

Attachments overview

There are several REST API endpoints that allow you to work with attachments of various Polarion objects like **Work Item** attachments, **Page** attachments, or **Document** attachments. Among other uses, they allow you to modify attachment fields and upload or download their actual content.

When retrieving the attachment content via one of the `content` endpoints, keep in mind that the actual attachment content is returned "as is", corresponding to the original octet stream previously uploaded to Polarion via the User Interface or the API. Polarion does not perform any validation, sanitization, or malware or virus scan when uploading or downloading attachment files.

The following sections show examples of how to create and update **Attachments**. The examples are equally applicable to any of the supported **Attachment** types, including the following:

Attachment type	REST API "type" value	Base URL example
Document Attachments	`document_attachments`	`https://myhost/polarion/rest/v1/projects/drivepilot/spaces/Requirements/documents/Project%20Scope/attachments`
Page Attachments	`page_attachments`	`https://myhost/polarion/rest/v1/projects/drivepilot/spaces/_default/pages/Dashboard/attachments`
Work Item Attachments	`workitem_attachments`	`https://myhost/polarion/rest/v1/projects/drivepilot/workitems/DP-1285/attachments`
Test Run Attachments	`testrun_attachments`	`https://myhost/polarion/rest/v1/projects/myProject/testruns/myTestRun/attachments`

Test Record Attachments	`testrecord_attachments`	`https://myhost/polarion/rest/v1/projects/myProject/testruns/myTestRun/testrecords/myProject/myTestCase/0/attachments`
Test Step Result Attachments	`teststepresult_attachments`	`https://myhost/polarion/rest/v1/projects/myProject/testruns/myTestRun/testrecords/myProject/myTestCase/0/teststepresults/1/attachments`

See the REST API Reference in the SDK documentation for the actual endpoint availability for the individual **Attachment** types.

Post multiple Attachments without LID

There are two ways to post **Attachments**: with or without using a local ID (LID). When LID is not used and there are multiple **Attachments** in the request, the **Attachments** are assigned based on their order of appearance in the request body.

Example

- **Example Request:**

  `POST https://myHost/polarion/rest/v1/projects/drivepilot/workitems/DP-1285/attachments`

- **Request Body:**

  Resource - (Object)

```
{
  "data": [
    {
      "type": "workitem_attachments",
      "attributes": {
        "fileName": "Everest",
        "title": "Everest"
        }
    },
    {
         "type": "workitem_attachments",
         "attributes": {
              "fileName": "science",
              "title": "science"
        }
    }
  ]
}
```

Files - (Array) - Files to post

- **Response Body:**

  201 - Created

```
{
    "data": [
      {
        "type": "workitem_attachments",
        "id": "drivepilot/DP-1285/5-Everest",
        "links": {
            "self": "https://myHost/rest/v1/projects/drivepilot/
workitems/DP-1285/attachments/5-Everest",
            "content": "https://myHost/rest/v1/projects/drivepilot/
workitems/DP-1285/attachments/5-Everest/content"
        }
    },
    {
        "type": "workitem_attachments",
        "id": "drivepilot/DP-1285/6-science",
        "links": {
            "self": "https://myHost/polarion/rest/v1/projects/
drivepilot/workitems/DP-1285/attachments/6-science",
            "content": "https://myHost/polarion/rest/v1/projects/
drivepilot/workitems/DP-1285/attachments/6-science/content"
      }
    }
  ]
}
```

Note

The above example also applies to posting or creating a single **Attachment**. The only difference in the payload is a single entry under the `data` section of the resource, followed by a single file under the `files` section of the request body. If the file name is passed as an empty string ("") or null in a `POST` request, then the passed Input stream file name will be used to set the file name attribute on the attachment.

Post multiple Attachments with LID

The alternative way of creating multiple **Attachments** with a single request is by using the local IDs (LIDs) to assign the content to the corresponding **Attachment** resources.

Example

- **Example Request:**

```
POST https://myHost/polarion/rest/v1/projects/drivepilot/workitems/
DP-1285/attachments
```

- **Request Body:**

  Resource - (Object)

```
{
    "data": [
     {
        "type": "workitem_attachments",
        "lid": "content_badami",
        "attributes": {
            "title": "badami",
        }
     },
     {
        "type": "workitem_attachments",
        "lid": "content_belur"
        "attributes": {
            "title": "belur",

        }
    }
  ]
}
```

  Files- (Files to post with LID as key)

  `content_badami : badami.jpg`

  `content_badami : belur.jpg`

- **Response Body:**

  201 - Created

```
{
    "data": [
      {
        "type": "workitem_attachments",
        "id": "drivepilot/DP-1285/5-badami.jpg",
        "links": {
            "self": "https://myHost/rest/v1/projects/drivepilot/
workitems/DP-1285/attachments/5-badami.jpg",
            "content": "https://myHost/rest/v1/projects/drivepilot/
workitems/DP-1285/attachments/5-badami.jpg/content"
        }
    },
      {
        "type": "workitem_attachments",
        "id": "drivepilot/DP-1285/6-belur.jpg",
        "links": {
            "self": "https://myHost/polarion/rest/v1/projects/
drivepilot/workitems/DP-1285/attachments/6-belur.jpg",
            "content": "https://myHost/polarion/rest/v1/projects/
drivepilot/workitems/DP-1285/attachments/6-belur.jpg/content"
        }
    }
  ]
}
```

Patch the content and title of an Attachment

Description: Updates the **Attachments** of a specified **Work Item**.

Example

- **Example Request:**

  `PATCH https://myHost/polarion/rest/v1/projects/drivepilot/workitems/DP-1285/attachments/7-everest.png`

- **Request Body:**

  Resource - (Object)

```
{
    "data": {
        "type": "workitem_attachments",
        "id": "drivepilot/DP-1285/7-everest.png",
        "attributes": {
            "title": "Updated"
        }
    }
}
```

  Content - (binary) (`science.jpg`)

- **Response Body:**

  204- No content

Insert a Document Attachment to the Document content

Attachments that were created for a **Document** using a `POST` request do not appear in the **Document's** content, but they are displayed in the **Attachments sidebar**. To add them to the **Document's** content, update the `homePageContent` field of the **Document** (by `PATCH` **Document** endpoint).

To update the **Document's** content, add an `<img src="attachment:attachment_id"/>` element to the desired position.

Note

The whole value of `homePageContent` needs to be sent in the request body, you cannot send the updated or inserted part only.

Example

- **Example Request:**

  `PATCH https://hostname/polarion/rest/v1/projects/{projectId}/spaces/{spaceId}/documents/{documentId}`

- **Request Body:**

```
{
  "data":{
    "type":"documents",
    "id":"projectId/spaceId/documentId",
    "attributes":{
      "homePageContent":{
        "type":"text/html",
        "value": "...<img src=\"attachment:myAttachment.pdf\" 
style=\"max-width: 250px;\"/>..."
      }
    }
  }
}
```

Managing users and setting a license

Create users

When creating users, the endpoint is `POST /users`.

In the following two examples, the newly created global user accounts return the `201` response after successfully creating a user.

Create a single user:

Example

- **Example Request:**

  `POST http://myHost/polarion/rest/v1/users`

- **Request Body:**

```
{
  "data": [
    {
      "type": "users",
      "attributes": {
        "initials": "RU",
        "name": "Rest, User",
        "id": "rest-user",
        "email": "rest.user@example.com",
        "disabledNotifications": true
      }
    }
  ]
}
```

- **Response Body:**

  201 - Created

```
{
  "data": [
    {
      "type": "users",
      "id": "rest-user",
      "links": {
        "self": "https://myHost/polarion/rest/v1/users/rest-user"
      }
    }
  ]
}
```

Create multiple users:

Example

- **Example Request:**

  `POST http://myHost/polarion/rest/v1/users`

- **Request Body:**

```
{
  "data": [
    {
      "type": "users",
      "attributes": {
        "initials": "RU1",
        "name": "Rest, User1",
        "id": "rest-user1",
        "email": "rest.user1@example.com",
        "disabledNotifications": true
      }
    },{
      "type": "users",
      "attributes": {
        "initials": "RU2",
        "name": "Rest, User2",
        "id": "rest-user2",
        "email": "rest.user2@example.com",
        "disabledNotifications": true
      }
    }
  ]
}
```

- **Response Body:**

  201 - Created

```
{
  "data": [
    {
      "type": "users",
       "id": "rest-user1",
       "links": {
         "self": "https://myHost/polarion/rest/v1/users/rest-user1"
       }
     },{
      "type": "users",
       "id": "rest-user1",
       "links": {
         "self": "https://myHost/polarion/rest/v1/users/rest-user2"
      }
    }
  ]
}
```

Set license

When setting licenses, the endpoint is `POST /users/{userId}/actions/setLicense`.

In the following examples, the Polarion license is assigned to the user. The `204` response is returned when successfully assigning a license to the user.

Note

You can assign a license to any user via the REST API as long as they do not already have one. When attempting to set a license to a user that already has one results in a `400 Bad Request` response.

Set a concurrent license:

Example

- **Example Request:**

  `POST https://myhost/polarion/rest/v1/users/admin/actions/setLicense`

- **Request Body:**

  ```
  {
    "license": "ALM",
    "group": "group name",
    "concurrent": true
  }
  ```

- **Response Body:**

  204 - No content

Set a named license:

Example

- **Example Request:**

  `POST https://myhost/polarion/rest/v1/users/admin/actions/setLicense`

- **Request Body:**

  ```
  {
    "license": "QA",
    "concurrent": false
  }
  ```

- **Response Body:**

  204 - No content.

User avatar

To fetch your user avatar use the `GET .../user/{ID}/actions/getAvatar` endpoint. If you do not have an avatar, the system returns the avatar of the default user.

To create and update your avatar, you can use the `POST .../user/{ID}/actions/updateAvatar` endpoint. If you send empty data, the avatar is cleared. The maximum allowed size for an avatar file is 50 KB. The file must be in one of the following file formats:

- `.jpg`
- `.png`
- `.jpeg`

- `.gif`

Perform workflow action

You can also change the workflow status of a Work Item via the REST API by using the `workflowAction` URL parameter in a PATCH request. The value of the parameter is the **ID** of the selected action defined in the Workflow configuration.

Perform Workflow action on a Work Item

Example

- **Example Request:**

  `PATCH http://myHost/polarion/rest/v1/projects/projectId/workitems/workItemId?workflowAction=start_progress`

- **Request Body (The same as a regular Work Item PATCH):**

```
{
    "data": {
        "type": "workitems",
        "id": "projectId/workItemId",
        "attributes": {
            "title": "Title after start_progress action"
        },
        "relationships": {
            "assignee": {
                "data": [
                  {
                    "id": "admin",
                    "type": "users"
                }
              ]
            }
        }
    }
  }
}
```

- All necessary data for the workflow action should be filled in. If some of it is missing, the action generates the following exception:

```
{
    "errors": [
      {
        "status": "400",
        "title": "Bad Request",
        "detail": "'Start Progress' failed for Work Item 'workItemId':
The required field 'assignee' of Work Item 'workItemId' is empty.",
          "source": {
              "parameter": "workflowAction"
          }
      }
    ]
}
```

- If the workflow change is not allowed based on the current status, the action generates the following exception:

```
{
    "errors": [
      {
          "status": "400",
          "title": "Bad Request",
          "detail": "The workflow action 'start_progress' was not found.",
          "source": {
              "parameter": "workflowAction"
          }
      }
    ]
}
```

Perform a Workflow action on a Document

Example

- **Example Request:**

  PATCH http://myHost/polarion/rest/v1/projects/projectId/spaces/spaceId/documents/documentId?workflowAction=publish

- **Request Body (The same as a regular Document PATCH):**

```
{
    "data": {
        "type": "documents",
        "id": "projectId/spaceId/documentId",
        "attributes": {
            "title": "Title after publish action"
        }
    }
}
```

Work Items

Watches

To update Watches for a **Work Item**, set the list of **Watches** in the **Relationship** section of a PATCH request.

To add a new user as a **Watcher**, the entire original list of **Watchers** must be set with the new added value.

Example

- **Example Request:**

  PATCH https://myhost/polarion/rest/v1/projects/projectId/workitems/workItemId

- **Request Body:**

```
{
    "data": {
        "type": "workitems",
        "id": "projectId/workItemId",
        "relationships": {
            "watches": {
                "data": [
                  {
                    "type": "users",
                    "id": "userId"
                }
              ]
            }
        }
    }
}
```

- To remove all **Watches**, leave the **Watches** array empty or use the `delete relationships` endpoint.

  Request Body:

```
{
    "data": {
        "type": "workitems",
        "id": "projectId/workItemId",
        "relationships": {
            "watches": {
                "data": []
            }
        }
    }
}
```

Test Steps

You can manage the Test Case Test Steps by using the `teststeps` **Work Item** endpoints.

The step **ID** is the index of the row of the Test Steps table. The index starts from `1` (the first row is `1`, not `0`).

Example

- **Fetching a single Test Step**

  Example Request:

  ```
  GET https://myhost/polarion/rest/v1/projects/projectId/workitems/
  workItemId/teststeps/2?fields[teststeps]=@basic
  ```

- **Response Body:**

```
{
  "data": {
    "type": "teststeps",
    "id": "projectId/workItemId/2",
    "attributes": {
      "index": "2",
      "keys": [
        "step",
        "description",
        "expectedResult"
      ],
      "values": [
        {
          "type": "text/html",
          "value": "step two"
        },
        {
          "type": "text/html",
          "value": "step description two"
        },
        {
          "type": "text/html",
          "value": "expected result two"
        }
      ]
    },
      "links": {
        "self": "http://myhost/polarion/rest/v1/projects/projectId/
workitems/workItemId/teststeps/2"
    }
```

```
  }
}
```

Example

- **Adding a Test Step Collection**

  Request:

  POST ..projects/projectId/workitems/workItemId/teststeps

- **Request Body:**

```
{
  "data": [
    {
      "type": "teststeps",
      "attributes": {
        "keys": [
          "Step", "Description", "Result"
        ],
        "values": [
          {
            "type": "text/html",
            "value": "step 1"
          },
          {
            "type": "text/html",
            "value": "description 1"
          },
          {
            "type": "text/html",
            "value": "result 1"
          }
        ]
      }
    },
    {
      "type": "teststeps",
      "attributes": {
        "keys": [
          "Step", "Description", "Result"
        ],
        "values": [
          {
            "type": "text/html",
            "value": "step 2"
          },
          {
            "type": "text/html",
            "value": "description 2"
          },
          {
            "type": "text/html",
            "value": "result 2"
          }
        ]
      }
    }
  ]
}
```

- **Response Body**

  `201 - Created`

```
{
  "data": [
    {
      "type": "teststeps",
      "id": "projectId/workItemId/1",
      "links": {
        "self": "http://myhost/polarion/rest/v1/projects/projectId/
workitems/workItemId/teststeps/1"
        }
    },
    {
      "type": "teststeps",
      "id": "projectId/workItemId/2",
      "links": {
        "self": "http://myhost/polarion/rest/v1/projects/projectId/
workitems/workItemId/teststeps/2"
      }
    }
  ]
}
```

Custom field table

You can manage the **Work Item table** fields (`table` type custom fields) by using the primary **Work Item** endpoint. There can be several table fields.

Example

- **Example Request:**

  `PATCH ..projects/projectId/workitems/workItemId`

- **Request Body:**

```
{
  "data": {
      "type": "workitems",
      "id": "projectId/workItemId",
    "attributes": {
      "CarTable": {
        "keys": [
          "carbrands",
          "carModels",
          "carYear"
        ],
        "rows": [
          {
            "values": [
            {
              "type": "text/html",
              "value": "mercedes"
            },
            {
              "type": "text/html",
              "value": "E250"
            },
            {
              "type": "text/html",
              "value": "2021"
            }
          ]
        },
        {
          "values": [
              {
                "type": "text/html",
                "value": "mercedes"
              },
              {
                "type": "text/html",
                "value": "e500"
              },
              {
                "type": "text/html",
                "value": "2012"
              }
            ]
          }
        ]
      }
    }
  }
}
```

Changing the type

To change the type of a **Work Item**, use the `patch` **Work Item** endpoint that specifies the `changeTypeTo` query parameter. If the new type requires you to fill in certain fields, you receive an error message that list the required fields.

DO NOT change the type of a **Work Item** and perform a workflow action at the same time.

Example

- **Example Request:**

  `PATCH /polarion/rest/v1/projects/MYPROJECT/workitems/workItemId?changeTypeTo=something`

- **Request Body:**

```
{
    "data": {
      "type": "workitems",
      "id": "MYPROJECT/workItemId",
      "attributes": {}
    }
}
```

Get workflow actions

You can use the `getWorkflowActions` **Work Item** action endpoint to get a list of all workflow actions for a **Work Item**. It returns the list of actions in accordance with the workflow assigned to the **Work Item** type in the project settings. The data includes a reason for unavailability of unavailable actions.

Example

- **Example Request:**

  `GET /rest/v1/projects/projectId/workitems/workItemId/actions/getWorkflowActions`

- **Response Body:**

```
{
    "links": {
        "self": "http://myhost/polarion/rest/v1/projects/projectId/
workitems/workItemId/actions/getWorkflowActions",
        "first": "http://myhost/polarion/rest/v1/projects/projectId/
workitems/workItemId/actions/getWorkflowActions?page%5Bnumber%5D=1",
        "last": "http://myhost/polarion/rest/v1/projects/projectId/
workitems/workItemId/actions/getWorkflowActions?page%5Bnumber%5D=1"
    },
    "data": [
        {
            "id": "6",
            "available": true,
            "isSignatureRequired": true,
            "isAddingSignature": true,
            "nativeActionId": "quickly_accept",
            "name": "Quickly Accept",
            "requiredFields": [
                "initialEstimate"
            ],
            "targetStatus": "accepted",
            "requiredRoles": [
                "admin"
            ]
        },
        {
            "id": "7",
            "available": true,
            "nativeActionId": "reject",
            "name": "Reject",
            "requiredFields": [
                "resolution"
            ],
            "targetStatus": "rejected"
        },
        {
            "id": "8",
            "nativeActionId": "reviewed",
            "name": "Review",
            "targetStatus": "reviewed",
            "unavailableReason": "Configured workflow condition
'FieldNonEmpty' is not met because Work Item field 'Remaining Estimate'
is empty. \nWorkflow transition cannot occur until this condition is
satisfied."
        }
    ]
}
```

Work Item type icon

The icon of a **Work Item** can be fetched in two steps:

1. Get the type of the **Work Item**.
2. Get the enumeration item with its icon.

Example

- **Example Request:**

  `GET https://myhost/polarion/rest/v1/projects/projectId/workitems/workItemId?fields[workitems]=type`

- **Response Body:**

```
{
  "links": {
    "self": "https://myhost/polarion/rest/v1/projects/projectId/
workitems/workItemId?fields%5Bworkitems%5D=type"   },
  "data": {
    "type": "workitems",
    "id": "projectId/workItemId",
    "attributes": {
      "type": "workpackage"
    }
  }
}
```

Example

- **Example Request:**

  `GET https://myhost/polarion/rest/v1/projects/projectId/enumerations/~/workitem-type/~`

- **Response Body:**

```
{
  "links": {
    "self": "https://myhost/polarion/rest/v1/projects/projectId/
enumerations/~/workitem-type/~"   },
    "data": {
      "type": "enumerations",
      "id": "projectId/~/workitem-type/~",
      "attributes": {
        "options": [
                {
          "id": "workpackage",
          "name": "Work Package",
          "color": "#F1ED92",
          "description": "A set of actions that need to be taken to
fulfill a goal.",
          "iconURL": "/polarion/icons/default/enums/type_userstory.gif"
        }
      ],
      "enumName": "workitem-type"
    },
      "links": {
      "self": "https://myhost/polarion/rest/v1/projects/UKDrivePilot/
enumerations/%7E/workitem-type/%7E"
    }
  }
}
```

Plans and Plan templates

You can manage **Plans** and **Plan** templates on a project level.

To only fetch **Plan** templates, you have to parametrize the `GET` request: `?templates=true.`

Create a Plan

Example

To create a **Plan** with a template, you have to specify the `relationships` parameter.

- **Example Request:**

  `POST https://myhost/polarion/rest/v1/projects/elibrary/plans`

- **Request Body:**

```
{
  "data": [
    {
      "type": "plans",
      "attributes": {
        "id" : "plan2",
        "isTemplate" : false
      },
      "relationships": {
        "template": {
          "data": {
            "type": "plans",
            "id": "elibrary/release"
          }
        }
      }
    }
  ]
}
```

Adding Planned Work Items

Example

Use the `workitems` relationship of the **Plan** to add a **Work Item** to the **Plan**.

- **Example Request:**

  `POST https://.../plans/{planId}/relationships/workItems`

- **Request Body:**

```
{
  "data": [
    { "type": "workitems", "id": "{projectId}/{workitemId}" },
    { "type": "workitems", "id": "{projectId}/{workitemId}" }
  ]
}
```

Project management

The REST layer provides endpoints to create, mark, move, unmark, or delete a project.

These actions can take a longer time, because they run as a Polarion Job (Asynchronous Request Reply pattern). The complete process is executed as a two-step process.

1. The action is started and you receive the `202 Accepted` response if the process starts successfully. The response also sends the `jobId` of the Polarion Job which is started, along with a link to the job log.
2. The state and status of the action has to be obtained through the `jobs/{jobId}` endpoint. If the job finished successfully, the job response contains a `project` relationship to the changed object.

Example

- **Example Request (start of creating a project):**

  `POST /projects/actions/createProject`

- **Request Body:**

```
{
  "projectId": "TEST",
  "location": "/TEST",
  "trackerPrefix":"TEST",
  "templateId": "alm_drivepilot_template"
}
```

- **Response Body:**

```
{
  "data": {
    "type": "jobs",
    "id": "b512ab11-0af928a3-2b7551ee-8b3ac7d2",
    "attributes": {
      "jobId": "b512ab11-0af928a3-2b7551ee-8b3ac7d2",
      "name": "Creating project TEST",
      "state": "RUNNING"
    },
    "links": {
      "self": "http://localhost:8888/polarion/rest/v1/jobs/
b512ab11-0af928a3-2b7551ee-8b3ac7d2",
      "log": "http://localhost:8888//polarion/job-
report%3FjobId=b512ab11-0af928a3-2b7551ee-8b3ac7d2"
    }
    }
  }
```

Example

- **Example Request (create a project job state and status):**

  ```
  GET /rest/v1/jobs/b512ab11-0af928a3-2b7551ee-8b3ac7d2?
  fields[jobs]=@all&fields[projects]=@all&include=project
  ```

- **Response Body:**

```
{
  "links": {
    "self": "http://localhost:8888/polarion/rest/v1/jobs/b512ab11-0af928a3-2b755
fields%5Bjobs%5D=%40all&fields%5Bprojects%5D=%40all&include=project"
  },
    "data": {
      "type": "jobs",
      "id": "b512ab11-0af928a3-2b7551ee-8b3ac7d2",
      "attributes": {
        "jobId": "b512ab11-0af928a3-2b7551ee-8b3ac7d2",
        "name": "Creating project TEST",
        "state": "FINISHED",
        "status": {
          "type": "OK"
        }
      },
      "relationships": {
        "project": {
          "data": {
            "type": "projects",
            "id": "TEST"
          }
        }
      },
      "links": {
        "self": "http://localhost:8888/polarion/rest/v1/jobs/b512ab11-0af928a3-2
        "log": "http://localhost:8888//polarion/job-report%3FjobId=b512ab11-0af9
      }
    },
      "included": [
        {
          "type": "projects",
          "id": "TEST",
          "attributes": {
            "id": "TEST",
            "name": "TEST",
            "description": {
              "type": "text/plain",
"value": "This model project demonstrates how an actual project might be set up
Model Project' template."
            },
            "active": true,
```

```
            "trackerPrefix": "MYP",            "icon": "/polarion/icons/default
drive-pilot.svg",
            "color": "#2A989B"
          },
            "links": {              "self": "http://localhost:8888/polarion/res

      }
    }
  ]
}
```

Work Items in Documents

The fact that a **Work Item** resides in a **Document** is represented by a relationship `module` pointing to the corresponding **Document** resource. If this relationship does not have a value, it is not present in the response when fetching the **Work Item** resource, which indicates that the **Work Item** exists outside of any **Document**.

Work Item action endpoints are provided to move **Work Items** into, out of, and between **Documents**.

Create a Work Item in a Document

Specifying the `module` relationship when creating a **Work Item** means that the new **Work Item** is created in the given **Document** if the authenticated user has the permission to modify that **Document** (`com.polarion.persistence.object.Module.modifyContent`).

Example

- **Example Request:**

```
POST https://myhost/polarion/rest/v1/projects/myProject/workitems
```

- **Request Body:**

```
{
  "data": [
    {
      "type": "workitems",
      "attributes": {
        ...
      },
      "relationships": {
        "module": {
          "data": {
            "type": "documents",
            "id": "myProject/mySpace/MyDocument"
          }
```

```
            },
            ...
         }
      }
   ]
}
```

The **Work Item** is created in the **Document's** recycle bin by default. To make it actually appear in the **Document's** content, a separate REST API call is needed to create the corresponding `workitem` Document Part. See Document Parts for more details.

The `module` relationship of a **Work Item** cannot be modified via a PATCH request. The following sections discuss how to move **Work Items** between **Documents**.

Move a Work Item from a Document

To move a **Work Item** from a **Document**, send the following request without a request body and any parameters:

```
POST  https://myhost/polarion/rest/v1/projects/{projectId}/workitems/
{workItemId}/actions/moveFromDocument
```

The move was successful if the `204 - No Content` response code is displayed.

Move a Work Item to a Document or between Documents

To move a **Work Item** to a **Document** (if it was not included in a **Document** previously), or to move a **Work Item** between **Documents**, send the following request and request body:

Example:

Example Request:

```
POST  https://myhost/polarion/rest/v1/projects/{projectId}/workitems/
{workItemId}/actions/moveToDocument
```

Request Body:

```
{
  "targetDocument": "myProjectId/mySpaceId/myDocumentId",
  "previousPart": "myProjectId/mySpaceId/myDocumentId/myPartId1",
  "nextPart": "myProjectId/mySpaceId/myDocumentId/myPartId2"
}
```

The `previousPart` and `nextPart` parameters are used to specify the position of the moved **Work Item** in the new **Document**, either by pointing to the part that it should either follow or precede. Only one of the `previousPart`/`nextPart` parameters can be specified, and it must be the ID of an existing part of the target **Document**. If you do not specify neither `previousPart` nor `nextPart`, then the **Work Item** part is added at the end of the target **Document**.

The move was successful if the `204 - No Content` response code is displayed.

Documents

Documents overview

Polarion REST API provides ways to read, create, and manipulate **Documents**.

Document content can be read, set, or updated using the `homePageContent` field of the `documents` resource, or by using the endpoints for working with the structured `document_parts`.

Creating Documents

The following is an example of a request to create a new **Document** in the default space of the `myProject` project:

Example

- **Example Request:**

  `POST /polarion/rest/v1/projects/myProject/spaces/_default/documents`

- **Request Body:**

```
{
  "data": [{
    "type": "documents",
    "attributes": {
      "moduleName": "myDocument", // the ID, required
      "title": "My Document",  // not required, derived from moduleName
if absent
      "type": "req_specification", // value required to enable custom
fields and workflow
      "structureLinkRole": "has_parent", // required
     "renderingLayouts": [{
          "type": "requirement",
          "label": "Default Requirement Layout",
          "layouter": "paragraph",
          "properties": [{
              "key": "fieldsAtEnd",
              "value": "severity,status"
```

```
            },
            {
              "key": "fieldsAtStart",
              "value": "id"
            }
          ]
        }],
      "usesOutlineNumbering": true,
      "outlineNumbering": {
          "prefix": "MYDOC"
       },
      "homePageContent": {
          "type": "text/html",
          "value": "<h1>My Document</h1>\n<p
id=\"polarion_template_1\">First paragraph.</p>\n"
       }
     }
  }]
}
```

The `renderingLayouts` value is not required, but we recommend you set it for new **Documents**, or add it to the **Document** resource later using PATCH so that the **Work Items** in those **Documents** are rendered as desired.

`paragraph` is the default value for the `layouter` field. Other possible values are as follows:

- `default`
- `paragraph`
- `section`
- `titleTestSteps`
- `titleDescTestSteps`

Document content

The `homePageContent` attribute specifies the content of the **Document** in the Polarion DLE HTML format. If this attribute is missing, REST API automatically creates a single empty paragraph ready to be updated with content via the User Interface or subsequent API calls.

Note

Explicitly sending `null` or `""` as the `homePageContent` may create a non-editable **Document**.

When creating or updating **Documents**, there is only limited validation of the provided `homePageContent` value. If there is some malformed markup or paragraph parts IDs are missing, the change can be saved without errors but it will cause problems when you read the **Document** by parts.

As a solution, either update the **Document** with a fixed `homePageContent` value, or alternatively open, edit, and save the **Document** using the Document-like Editor in the Polarion User Interface.

Example

Using PATCH to update the Document content:

```
PATCH https://hostname/polarion/rest/v1/projects/{projectId}/spaces/
{spaceId}/documents/{documentId}
```

- **Example Request:**

```
{
   "data":{
      "type":"documents",
      "id":"projectId/spaceId/documentId",
      "attributes":{
         "homePageContent":{
            "type":"text/html",
            "value":"<p id=\"polarion_template_0\">This is a text</p><p
id=\"polarion_1\"><span style=\"font-size: 10pt;\">Welcome to <span
style=\"font-weight: bold;\">Polarion Rest API</span></span></p>"
         }
      }
   }
}
```

- **Request Body:**

```
204 - no content
```

Create and update Headings as part of the Document content

When updating a **Document's** `homePageContent`, you can include HTML elements that represent headings and their text. Such HTML parts are automatically converted to Heading **Work Items** when you save them.

Example

Create and update Headings as part of the Document content:

```
PATCH https://hostname/polarion/rest/v1/projects/{projectId}/spaces/
{spaceId}/documents/{documentId}
```

- **Example Request:**

```
{
   "data":{
      "type":"documents",
      "id":"projectId/spaceId/documentId",
      "attributes":{
         "homePageContent":{
            "type":"text/html",
            "value":"<h1>Title</h1><h2>Heading 1</h2><div
class=\"heading-12\">Heading level 11</div>"
         }
      }
   }
}
```

- **Request Body:**

```
204 - no content
```

Document Parts

A Document Part is the object representation of a top-level HTML node of the **Document** content. There are different Document Part types that represent different elements inside the **Document**:

Document element	HTML example	Type	REST ID example

Heading	`<h1 id="polarion_wiki macro name=module-workitem;params=id =DPP-100"></h1>`
	
	OR (for headings bigger than h6)
	
	`<div class="heading-7" id="polarion_wiki macro name=module-workitem;params=id =DPP-100"></div>`
		heading	`heading_DPP-100`

Work Item (**Document Work Item**)	`<div id="polarion_wiki macro name=module-`	workitem	`workitem_DPP-120`

	`workitem;params=id=DPP-120"></div>`		
Work Item (Referenced)	`<div id="polarion_wiki macro name=module-workitem;params=id=DPP-120|external=true"></div>`	workitem	`workitem_DPP-121`
Work Item (Referenced from different project)	`<div id="polarion_wiki macro name=module-workitem;params=id=DPP-123|external=true|project=drivepilot"></div>`	workitem	`workitem_DPP-123_project_drivepilot`
Wiki block	`<div id="polarion_editor_html_block46" class="polarion-dle-wiki-block"> ... </div>`	wikiblock	`wikiblock_46`
Page break	`<div id="polarion_wiki macro name=page_break;params=uid=6" contentEditable="false" data-is-landscape="false"></div>`	pagebreak	`pagebreak_6`
Table of contents	`<div id="polarion_wiki macro name=toc"></div>`	toc	`toc`
Table of figures	`<div id="polarion_wiki macro name=tof;params=uid=43"></div>`	tof	`tof_43`
Table	`<table id="polarion_wiki macro`	table	`table_2133`

	`name=table;params= uid=2133" class="polarion-Document-table" style="..."> <tbody> ... <tbody> </table>`		
Other (paragraphs, lists, and so on)	`<p id="polarion_416"> ... </p>`	normal	`polarion_416`

The following Document Parts-related operations are supported:

- **GET List of Document Parts** - returns **Document** `homePageContent` as a list of Document Parts elements.
- **GET Document Part** - returns specified Document Part by element ID.
- **POST Work Item as Document Part** - inserts **Work Item** from the **Document** recycle bin to a right place in the **Document**.
- **POST external Work item as Document Part** - Inserts an external **Work Item** into the intended place in the **Document**.

Post Work Item as Document Part

To POST a **Work Item** as a **Document Part** send the following request:

Example

Post Work Item as Document Part:

```
POST https://hostname/polarion/rest/v1/projects/{projectId}/spaces/
{spaceId}/documents/{documentId}/parts
```

- **Example Request:**

```
{
  "data": [
    {
      "type": "document_parts",
      "attributes": {
        "type": "workitem"
      },
      "relationships": {
        "nextPart": {
          "data": {
            "type": "document_parts",
            "id": "projectId/spaceId/documentId/nextPartId"
          }
        },
        "previousPart": {
          "data": {
            "type": "document_parts",
            "id": "projectId/spaceId/documentId/previousPartId"
          }
        },
        "workItem": {
          "data": {
            "type": "workitems",
            "id": "projectId/workItemId"
          }
        }
      }
    }
  ]
}
```

Note

Only one of the `previousPart/nextPart` fields can be specified. If `previousPart/nextPart` is not specified, the **Work Item** is inserted at the end of the **Document** content.

The specified **Work Item** already has to exist and be present in the **Document's** recycle bin. See **Create a Work Item in a Document** for more information.

- **Response Body:**

```
{
  "data": [
    {
      "type": "document_parts",
      "id": "projectId/spaceId/documentId/partId",
      "links": {
        "self": "https://hostname/polarion/rest/v1/projects/projectId/
spaces/spaceId/documents/documentId/parts/documentPartId"
      }
    }
  ]
}
```

POST external Work Item as Document Part:

External **Work Items**, items not part of the **Document** and have no module relationship, can be added as referenced **Work items**.

To POST an external Work Item as a Document Part send the following request:

Example Request:

```
POST https://hostname/polarion/rest/v1/projects/{projectId}/spaces/
{spaceId}/documents/{documentId}/parts
```

Request Body:

```
{
  "data": [
   {
     "type": "document_parts",
     "attributes": {
      "type": "workitem"
     },
     "relationships": {
       "nextPart": {
        "data": {
         "type": "document_parts",
         "id": "projectId/spaceId/documentId/nextPartId"
        }
      },
      "previousPart": {
        "data": {
         "type": "document_parts",
         "id": "projectId/spaceId/documentId/previousPartId"
        }
      },
      "workItem": {
        "data": {
         "type": "workitems",
         "id": "projectId/workItemId"
        }
       }
      }
     }
    ]
   }
```

Response Body:

```
{
   "data": [
     {
       "type": "document_parts",
       "id": "projectId/spaceId/documentId/partId",
       "links": {
         "self":
"https://hostname/polarion/rest/v1/projects/projectId/spaces/spaceId/
documents/documentId/parts/documentPartId"
        }
       }
```

```
      ]
    }
```

POST Document Part with level

The POST Document Parts endpoint provides the possibility to specify the indent level for the new Document Part. The default level is 0.

You can change the level of an already existing Document Part by PATCHing the **Document** `homePageContent`.

Example

Example Request:

```
POST https://hostname/polarion/rest/v1/projects/{projectId}/spaces/
{spaceId}/documents/{documentId}/parts
```

- **Request Body:**

```
{
  "data": [
    {
      "type": "document_parts",
      "level": 1,
      "attributes": {
        "type": "workitem"
      },
      "relationships": {
         "previousPart": {
          "data": {
            "type": "document_parts",
            "id": "projectId/spaceId/documentId/previousPartId"
          }
        },

        "workItem": {
          "data": {
            "type": "workitems",
            "id": "projectId/workItemId"
          }
        }
      }
    }
  ]
}
```

- **Response Body:**

```
{
  "data": [
    {
      "type": "document_parts",
      "id": "projectId/spaceId/documentId/partId",
      "links": {
        "self": "https://hostname/polarion/rest/v1/projects/projectId/
spaces/spaceId/documents/documentId/parts/documentPartId"
      }
    }
  ]
}
```

Document comments

You can fetch, update (resolve or reopen), and create **Document** comments via **Document** Comments endpoints. You can create `Replies` comments by a `POST` endpoint specifying the `parentComment` relationship. Without that relationship, a top-level comment is created. A new top-level comment created by the REST API appears as `Unreferenced`, meaning that it does not appear in the **Document's** content. To add it to the content, also update the `homePageContent` field of the **Document** with the `PATCH` **Document** endpoint.

For example, the comment is created first:

Example

- **Example Request:**

  `POST https://hostname/polarion/rest/v1/projects/{projectId}/spaces/{spaceId}/documents/{documentId}/comments`

- **Request Body:**

```
{
  "data": [
    {
      "type": "document_comments",
      "attributes": {
        "text": {
          "type": "text/html",
          "value": "My text value"
        }
      },
      "relationships": {
        "author": {
          "data": {
            "type": "users",
            "id": "userId"
          }
        }
      }
    }
  ]
}
```

And then the **Document's** content is updated by adding a `<span id="polarion-comment:created_comment_id"></span>` element to the desired position.

Note

The whole value of `homePageContent` needs to be sent in the request body, you cannot send the updated or inserted part only.

Example

- **Example Request:**

  `PATCH https://hostname/polarion/rest/v1/projects/{projectId}/spaces/{spaceId}/documents/{documentId}`

- **Request Body:**

```
{
  "data":{
     "type":"documents",
     "id":"projectId/spaceId/documentId",
     "attributes":{
        "homePageContent":{
           "type":"text/html",
           "value":"...<span id=\"polarion-comment:1\"></span>..."
        }
      }
   }
}
```

Note

If the new top-level comment is to be placed in some of the **Document's Work Items**, the description of the corresponding **Work Item** resource has to be updated with the same `span` element instead of **Document's** content.

Branch Documents

Branch a single Document

To branch a **Document**, use the **Document's** resource URL and send a `POST` request with a special request body like the example below. You can include the `revision` URL parameter to create a from a specific revision. Without the `revision` parameter, it will branch from the HEAD. You can also include `query` in the request body field to specify the **Work Items** to branch.

Example

- **Example Request:**

  `POST https://myhost/polarion/rest/v1/projects/myProject/spaces/mySpace/documents/MyDocument/actions/branch?revision=2345`

- **Request Body:**

```
{
    "targetProjectId": "myOtherProject",
    "targetSpaceId": "myOtherSpace",
    "targetDocumentName": "MyDocumentBranched",
    "copyWorkflowStatusAndSignatures": false,
    "query": "status:draft"
}
```

- **Response Body:**

```
{
    "data": {
        "type": "documents",
        "id": "myOtherProject/myOtherSpace/MyDocumentBranched",
        "attributes": {...},
        "relationships": {...},
        "links": {
            "self": "https://myhost/polarion/rest/v1/projects/
myOtherProject/spaces/myOtherSpace/documents/MyDocumentBranched"
        }
    }
}
```

Branch several Documents at once

To branch several **Documents** at once, use the `/all/documents/actions/branch` URL and send a `POST` request with a special Request Body like in the example below. This action is executed asynchronously, so in the response you receive the ID of a job that was started to perform the task. You can then use another `/jobs/{jobId}` endpoint to get information about the job's status. If the job finished successfully, the job response contains a `documents` relationship to all the branched **Documents**.

Example

- **Example Request:**

  `POST https://myhost/polarion/rest/v1/all/documents/actions/branch`

- **Request Body:**

```
{
  "documentConfigurations": [
    {
      "sourceDocument": "myProject/mySpace/myDocument1",
      "targetDocumentName": "myDocument1_branched"
    },
    {
      "sourceDocument": "myProject/mySpace/myDocument2",
      "targetDocumentName": "myDocument2_branched"
    }
  ]
}
```

- **Response Body:**

```
{
  "data": {
    "type": "jobs",
    "id": "6fb118b2-927aec93-13c0ef19-91ebc29f",
    "attributes": {
      "jobId": "6fb118b2-927aec93-13c0ef19-91ebc29f",
      "name": "Branch Documents",
      "state": "RUNNING"     },
    "links": {...}
  }
}
```

Example

- **Example Request:**

  ```
  GET https://myhost/polarion/rest/v1/jobs/
  6fb118b2-927aec93-13c0ef19-91ebc29f?fields[jobs]=@all
  ```

- **Response Body:**

```
{
  "links": {...},
  "data": {
    "type": "jobs",
    "id": "6fb118b2-927aec93-13c0ef19-91ebc29f",
    "attributes": {
      "jobId": "6fb118b2-927aec93-13c0ef19-91ebc29f",
      "name": "Branch Documents",
```

```
        "state": "FINISHED",
        "status": {
          "type": "OK"
        }
      },
      "relationships": {
        "documents": {
          "data": [
            {
              "type": "documents",
              "id": "myProject/mySpace/myDocument1_branched"
            },
            {
              "type": "documents",
              "id": "myProject/mySpace/myDocument2_branched"
            }
          ]
        }
      },
      "links": {...}
    }
  }
```

Copy a Document

To copy a **Document**, use the **Document's** resource URL and send a POST request with a special request body like the example below. You can include the `revision` URL parameter to create a copy from a specific revision. Without the revision parameter, it will copy the HEAD revision.

Example

- **Example Request:**

  `POST https://myhost/polarion/rest/v1/projects/myProject/spaces/mySpace/documents/MyDocument/actions/copy?revision=2345`

- **Request Body:**

```
{
    "targetProjectId": "myOtherProject",
    "targetSpaceId": "myOtherSpace",
    "targetDocumentName": "MyDocumentCopied",
    "removeOutgoingLinks": true,
    "linkOriginalItemsWithRole": "duplicates"
}
```

- **Response Body:**

```
{
    "data": {
        "type": "documents",
        "id": "myOtherProject/myOtherSpace/MyDocumentCopied",
        "attributes": {...},
        "relationships": {...},
        "links": {
            "self": "https://myhost/polarion/rest/v1/projects/
myOtherProject/spaces/myOtherSpace/documents/MyDocumentCopied"
        }
    }
}
```

Merge Documents

To merge **Work Items** between **Branched** and **Master Documents**, use the **Branch Document's** resource URL and send a POST request with a special request body like the example below:

This action is executed asynchronously, so you receive the **ID** of a job that was started to perform the task in the response. You can then use the `/jobs/{jobId}` endpoint to get information about the job's status.

If the job finishes successfully, the job response contains a document relationship to the **Target Document** of the merge.

If the merge finished with some changes in the **Target Document**, the relationship in the response contains the revision number created by the merge changes.

Example Request:

`POST`

`https://myhost/polarion/rest/v1/projects/myProject/spaces/mySpace/documents/MyDocument/actions/mergeToMaster`

`https://myhost/polarion/rest/v1/projects/myProject/spaces/mySpace/documents/MyDocument/actions/mergeFromMaster`

Request Body:

```
{

  "createBaseline": true,
  "userFilter": "status:open"

}
```

Response Body:

```
{
  "data": {
    "type": "jobs",
    "id": "6fb118b2-927aec93-13c0ef19-91ebc29f",
    "attributes": {
      "jobId": "6fb118b2-927aec93-13c0ef19-91ebc29f",
      "name": "Automatic Merge Documents",
      "state": "RUNNING"
     },
     "links": {...}
  }
}
```

Example Request:

```
GET https://myhost/polarion/rest/v1/jobs/6fb118b2-927aec93-13c0ef19-91ebc29f?
fields[jobs]=@all
```

Responses:

A Successful job: A new revision is created for the Target Document as a result of the Automatic Merge Job:

```
{
  "data": {
    "type": "jobs",
    "id": "6fb118b2-927aec93-13c0ef19-91ebc29f",
    "attributes": {
    "jobId": "6fb118b2-927aec93-13c0ef19-91ebc29f",
    "name": "Automatic Merge Documents",
    "state": "FINISHED",
    "status": {
      "type": "OK"
    }
  },
  "relationships": {
    "document": {
      "data": {
        "type": "documents",
        "id": "MyProjectId/MySpaceId/TargetDocumentId",
        "revision": "example"
      }
    }
   },
   "links": {
     "self": "https://localhost:8888/polarion/rest/v1/jobs/
6fb118b2-927aec93-13c0ef19-91ebc29f",
     "log": "https://localhost:8888/polarion/job-report?jobId=MyJobId"
     }
    }
}
```

A Successful job: No changes in the Target Document as a result of the Automatic Merge Job:

```
{
    "data": {
      "type": "jobs",
      "id": "6fb118b2-927aec93-13c0ef19-91ebc29f",
      "attributes": {
        "jobId": "6fb118b2-927aec93-13c0ef19-91ebc29f",
        "name": "Automatic Merge Documents",
        "state": "FINISHED",
        "status": {
          "type": "OK"}
        },
        "relationships": {
          "document": {
            "data": {
              "type": "documents",
              "id": "MyProjectId/MySpaceId/TargetDocumentId",
            }
          }
        },
        "links": {
          "self": "https://localhost:8888/polarion/rest/v1/jobs/MyJobId",
          "log": "https://localhost:8888/polarion/job-report?
jobId=6fb118b2-927aec93-13c0ef19-91ebc29f"
        }
     }
}
```

Enumerations

Enumeration resources are identified by `project ID`, enumeration context (**Document**, **Plans**, **Testing** or no context), enumeration name, and target type.

Enumeration context and enumeration name together (separated by "/", if there is a context) form the enumeration **ID** known from Java API.

See the table below for examples.

Enumeration ID (Java API)	Enumeration Context	Enumeration Name	Target Type	Note
severity		severity	testcase	A testcase-specific enumeration of severities.

documents/ document-status	documents	document-status		A general (not specific to any **Document** type) configuration of **Document** statuses.
@document		@document		An object enumeration of **Documents**.

Note

An empty enumeration context or empty target type is represented in the REST API request URLs by the ~ path parameter.

(Because the path parameters cannot be empty.)

In the following sections we describe operations related to enumerations:

- CRUD operations for the enumerations
- **Get options related to enumeration fields**
- **Read available enumeration icons**

CRUD operations for the enumerations

CRUD operations are supported for enumerations, however, not all enumerations support all CRUD operations. Some enumerations are not configurable at all - there are constant enumerations (for example `approval-status`) or object enumerations (for example `@document`). Such configurations can only be read.

Options that comprise the enumeration are stored in an `options` attribute (so options themselves are not resources), see the example of a GET response below.

Note

The read enumeration endpoints may return an inherited enumeration, for example an enumeration that does not exist for the exact specified project or target type, but is inherited from the global level or from a general enumeration.

Such an inherited configuration cannot be modified (`PATCH`) or deleted (`DELETE`). If you want to override the inherited configuration, you need to create (`POST`) a new configuration for the desired project or target type.

Example

CRUD operations for the enumerations:

```
GET https://myhost/polarion/rest/v1/projects/myProject/enumerations/~/
severity/testcase
```

- **Response Body:**

```
{
  "links": {...},
  "data": {
    "type": "enumerations",
    "id": "drivepilot/~/severity/testcase",
    "attributes": {
      "options": [
        {
          "id": "transition",
          "name": "Transition",
          "color": "#E74C3C",
          "iconURL": "/polarion/icons/default/enums/
severity_critical.gif"
        },
        {
          "id": "smoke",
          "name": "Smoke",
          "color": "#C0392B",
          "iconURL": "/polarion/icons/default/enums/severity_major.gif"
        },

        ...
      ],
      "enumName": "severity",
      "targetType": "testcase"
    },
    "links": {...}
  }
}
```

Get options related to enumeration fields

JAVA API fields of enumeration type are available in the REST API as either attributes or relationships (relationships are used for object-based enumerations). For every such field the REST API provides actions to get the current value of the field as a list of options, or to get available options that can be set to the field. In the following sections we will take a look at these actions in more detail.

Note

The enumeration field actions are supported even for object-based enumeration fields of JAVA API that are not yet supported as relationships, so they actually provide the only way to get their value using the REST API.

Get current options of an enumeration field

When a resource has an enumeration attribute (for example `severity` attribute of the `workitems` resource), the value of the attribute contains only the ID of the enumeration option (for example `critical`). To get the corresponding enumeration option with all its details, you can use the `getCurrentOptions` action available per field.

Note

The response always contains a list of options. For a single valued field this list has one item at most.

Example

Get current options of an enumeration field:

```
GET https://myhost/polarion/rest/v1/projects/myProject/workitems/WI-123/
fields/severity/actions/getCurrentOptions
```

- **Response Body:**

```
{
  "links": {...},
  "data": [
    {
      "id": "severity",
      "name": "Critical",
      "color": "#C0392B",
      "iconURL": "/polarion/icons/default/enums/severity_critical.gif"
    }
  ]
}
```

Get available options of an enumeration field

For any enumeration field there is a `getAvailableOptions` action that returns the options fot the corresponding enumeration. This action is provided for a particular resource instance (for example for a particular `workitems` resource), and also independently of any resource instance of a given target type (for example for `workitems` of `testcase` type). The latter case is useful to get the available options when creating a new resource of a given type.

The response format is exactly the same as in the case of `getCurrentOptions`, so we will not repeat it in the examples below.

Example

```
GET https://myhost/polarion/rest/v1/projects/myProject/workitems/WI-123/
fields/severity/actions/getAvailableOptions
```

Returns severity options available for the WI-123 **Work Item**. If the type of the **Work Item** is `testcase`, the options are found in the testcase-specific enumeration of severities (or in inherited enumeration, if that specific one is not configured).

```
GET https://myhost/polarion/rest/v1/projects/myProject/workitems/fields/
severity/actions/getAvailableOptions?type=testcase
```

Returns severity options available for **Work Items** of the `testcase` type in the `myProject` project. The options are found in the testcase-specific enumeration of severities (or in inherited enumeration, if that specific one is not configured).

Read available enumeration icons

Available enumeration icons are also resources in the REST API. There are endpoints to get the default icons and custom icons (which can be configured per project or globally).

Example

Return all default icons:

```
GET https://myhost/polarion/rest/v1/enumerations/defaulticons?
fields[icons]=@all
```

- **Response Body:**

```
{
  "links": {...},
  "data": [
    {
      "type": "icons",
      "id": "default/CMMI_stakeholders.gif",
      "attributes": {
        "path": "/polarion/icons/default/enums/CMMI_stakeholders.gif",
        "iconUrl": "https://myhost/polarion/icons/default/enums/
CMMI_stakeholders.gif"
      },
      "links": {...}
    },
    ...
 ],
  "meta": {...}
}
```

Create enumeration icons

You can create icons for enumerations in the Project or Global context.

Note

You cannot create default icons. The size of icons cannot exceed 16x16 pixels.

Example

Create enumeration icons:

```
POST https://myhost/polarion/rest/v1/enumerations/icons
```

- **Request Body (resource object):**

```
{
  "data": [
    {
      "type": "icons"
    },
    {
      "type": "icons"
    }
  ]
}
files- (Array) - twitter.png, facebook.png (Icon files)
```

- **Response Body:**

```
{
  "data": [
    {
      "type": "icons",
      "id": "group/facebook.png",
      "links": {
        "self": "https://myhost/polarion/rest/v1/enumerations/icons/
facebook.png"
      }
    },
    {
      "type": "icons",
      "id": "group/twitter.png",
      "links": {
        "self": "https:/myhost/polarion/rest/v1/enumerations/icons/
twitter.png"
```

```
         }
      }
   ]
}
```

Custom Enumeration Object Factories

Custom Object Enumerations are supported via Custom Object Enumeration factories. Plugins providing Custom Object Enumerations must implement `com.polarion.platform.persistence.IEnumObjectFactory.getEnumOptionPrototypes()`.

- If a custom plug-in that supplies the object factories implements `getEnumOptionPrototypes`, then the field that uses the custom object enumeration is displayed as an enum relationship.
- Otherwise, it is displayed as a string (the default). The default display is a list of strings if the custom field is a multi-value enum type.

Response Body (GET): For a custom field where **custom_testruntemplate_single** is supplied but prototype information is not:

```
{
   "links":{
      "self":"http://localhost:8888/polarion/rest/v1/projects/base/
workitems/WI-5?fields%5Bworkitems%5D=%40all"
   },
   "data":{
      "type":"workitems",
      "id":"base/WI-5",
      "attributes": {
         "id":"WI-5",
         "type":"task",
         "title":"Title before patch action.",
         "description": {
            "type":"text/plain",
            "value":"plain text description"
         },
         "severity":"normal",
         "priority":"50.0",
         "status":"open",
         "created":"2024-02-27T14:31:28.240Z",
         "updated":"2024-02-27T14:32:03.762Z",
         "custom_plantemplate_single":"base/testPlanTemplate"
      },
      "relationships":{
         "project":{
            "data":{
              "type":"projects",
              "id":"base"
            },
            "links":{
               "self":"http://localhost:8888/polarion/rest/v1/projects/base/
workitems/WI-5/relationships/project"
            }
         },
         "author": {
            "data": {
               "type":"users",
               "id":"jUnitTester"
            },
            "links":{
               "self":"http://localhost:8888/polarion/rest/v1/projects/base/
workitems/WI-5/relationships/author"
            }
         }
      },
      "links":{
         "self":"http://localhost:8888/polarion/rest/v1/projects/base/
workitems/WI-5",
         "portal":"http://localhost:8888/polarion/redirect/project/base/
workitem?id=WI-5""   "
      }
   }
}
```

Response Body (GET): For a custom field where both **custom_testruntemplate_single** and prototype information are supplied:

```
{
   "links":{
      "self":"http://localhost:8888/polarion/rest/v1/projects/base/
workitems/WI-6?fields%5Bworkitems%5D=%40all"" "
   },
   "data":{
      "type":"workitems",
      "id":"base/WI-6",
      "attributes": {
         "id":"WI-6",
         "type":"task",
         "title":"Title before patch action.",
         "description": {
           "type":"text/plain",
           "value":"plain text description"
         },
         "severity":"normal",
         "priority":"50.0",
         "status":"open",
         "created":"2024-02-27T14:46:43.550Z",
         "updated":"2024-02-27T14:47:27.335Z"
      },
      "relationships":{
         "project":{
            "data":{
               "type":"projects",
               "id":"base"
            },
            "links":{
               "self":"http://localhost:8888/polarion/rest/v1/projects/base/
workitems/WI-6/relationships/project"
            }
         },
         "author":{
            "data":{
               "type":"users",
               "id":"jUnitTester"
            },
            "links":{
               "self":"http://localhost:8888/polarion/rest/v1/projects/base/
workitems/WI-6/relationships/author"
            }
         },
         "assignee":{
            "links":{
               "self":"http://localhost:8888/polarion/rest/v1/projects/base/
workitems/WI-6/relationships/assignee"
            }
         },
         "custom_testruntemplate_single": {
            "data":{
               "type":"testruns",
               "id":"base/testRunTemplate"
            },
            "links":{
               "self":"http://localhost:8888/polarion/rest/v1/projects/base/
workitems/WI-6/relationships/custom_testruntemplate_single"
            }
         }
      },
      "links":{
         "self":"http://localhost:8888/polarion/rest/v1/projects/base/
workitems/WI-6",
         "portal":"http://localhost:8888/polarion/redirect/project/base/
workitem?id=WI-6"
      }
   }
}
```

If the custom object enumeration factory plug-in does not implement `getEnumOptionPrototypes`, then the payload must be supplied as a custom attribute for the custom enum option/custom enum options field, and the object is created/updated with the enum option(s).

Request Body (POST): if both a custom field **custom_plantemplate_single** and prototype information are NOT supplied:

```
{
   "data":[
      {
         "type":"workitems",
         "attributes":{
            "type":"task",
            "title":"WI-custom-single-enum-option-001",
            "description":{
               "type":"text/html",
               "value":"<b>rich description</b>"
            },
            "custom_plantemplate_single":"base/testPlanTemplate",
            "severity":"must_have",
            "priority":"71.0",
            "status":"closed",
            "resolution":"done",
            "initialEstimate":"3d",
            "timeSpent":"1d",
            "remainingEstimate":"2d",
            "dueDate":"2022-10-24"
         }
      }
   ]
}
```

If the custom object enumeration factory plug-in implements the `getEnumOptionPrototypes`, then the payload must be supplied as a relationship object for the custom enum options field and the object is created/updated with the enum option(s).

Request Body (POST): if a custom field **custom_plantemplate_single** and prototypes information is not supplied.

```
{
   "data":[
      {
         "type":"workitems",
         "attributes":{
           "type":"task",
           "title":"WI-custom-single-enum-option-relship-001"
         },
         "relationships": {
            "assignee":{
               "data":[
                 {
                    "type":"users",
                    "id":"user1"
                  }
               ]
            },
            "author":{
              "data":{
                  "type":"users",
                  "id":"user1"
               }
            },
            "custom_testruntemplate_single": {
               "data":{
```

```
                        "type":"testruns",
                        "id":"base/testRunTemplate"
                    }
                }
            }
        }
    ]
}
```

Test Management

Test Management encompasses the artifacts that are instrumental in providing testing-related functionality. Endpoints are provided for the following artifacts:

- Test Run
- Test Run template
- Test Run attachment
- Test Run comment
- Test Parameter definition
- Test Parameter
- Test Record
- Test Record attachment
- Test Step Result
- Test Step Result attachment

The endpoint for fetching Test Runs and Test Run templates is the same. The value of the `templates` query parameter decides whether to fetch Test Run templates or Test Runs.

Test Records

In Polarion, Test Records are created as a result of the execution of Test Cases. The execution action performs an automatic verification, creates artifacts (for example **Defects**), and relates them in the background. The execution action is currently not supported on the REST endpoints, so you have to use multiple endpoints to execute and create executed Test Records.

For example, if the **Defect** that is created when a Test Case fails (since the system is configured to automatically create a **Defect** if a Test Record fails), the **Defect** has to be created manually and specified as input relation while creating Test Records.

Test Records are zero indexed on the REST layer.

You have to specify the `testCaseRevision` parameter if you have to create or update a Test Record with a specific revision of the Test Case.

Test execution using signed context is not yet supported on the REST layer.

Example

Create Test Records

- **Example Request:**

  `POST https://myhost/polarion/rest/v1/projects/myProject/testruns/myTestRun/testrecords`

- **Request Body:**

```
{
  "data": [
    {
      "type": "testrecords",
      "attributes": {
        "comment": {
          "type": "text/html",
          "value": "my comment"
        },
        "duration": 0,
        "executed": "2023-01-01T00:00:00Z",
        "result": "failed"
      },
      "relationships": {
        "defect": {
          "data": {
            "type": "workitems",
            "id": "myProject/myDefectId"
          }
        },
        "executedBy": {
          "data": {
            "type": "users",
            "id": "MyUserId"
          }
        },
        "testCase": {
          "data": {
            "type": "workitems",
            "id": "myProject/myTestCase"
          }
        }
      }
    }
  ]
}
```

- **Response Body:**

```
{
  "data": [
    {
      "type": "testrecords",
      "id": "myProject/myTestRun/myProject/myTestcase/0",
      "links": {
        "self": "https://myHost/polarion/rest/v1/projects/myProject/
testruns/myTestRun/testrecords/myProject/myTestcase/0"
      }
    }
  ]
}
```

Test Record Deletion

Test Records can be deleted via the REST API. Only a single Test Record can be deleted at a time.

Example

Request:

http://localhost:8888/polarion/rest/v1/projects/ProjectID/testruns/TestRunIde/testrecords/TestCaseProject/TestCaseId/0

Response Body:

```
{
  "errors": [
    {
      "status": "404",
      "title": "Not Found",
      "detail": "Test Record 'ProjectID/TestRunIde/TestCaseProject/
TestCaseId/0' was not found.",
      "source": {
        "pointer": null,
        "parameter": null,
        "resource": {
          "id": "ProjectID/TestRunIde/TestCaseProject/TestCaseId/0",
          "type": "testrecords"
        }
      }
    }
  ]
}
```

Test Parameter Definition

Test Parameter Definitions define the name of the parameter. Test Parameter Definitions can be added to a Library, Project, and also to Test Runs.

Test Parameter

The value added to a Test Parameter Definition is the Test Parameter. Test Parameters can be added to Test Runs and Test Records.

Import Test Results

The REST layer provides endpoints to import Test Results in XUnit and Excel formats. Since the import process can take a long time to complete, the process is run as a Polarion Job (Asynchronous Request Reply pattern). The complete process is executed as a two-step process:

1. The import process is started and you receive the `202 Accepted` response if the process started successfully. The response also sends the `jobId` of the Polarion Job which is started, along with a link to the job log.
2. The state and status of the import process has to be obtained through the `jobs/{jobId}` endpoint.

Example

- **Example Request (beginning of import):**

  ```
  POST https://myhost/polarion/rest/v1/projects/myProject/testruns/
  myTestRun/actions/importXUnitTestResults
  ```

- **Response Body (202 Accepted):**

```
{
  "data": {
    "type": "jobs",
    "id": "8dd9cae6-927aec93-66855dab-098a0106",
    "attributes": {
      "jobId": "8dd9cae6-927aec93-66855dab-098a0106",
      "name": "Import of test results",
      "state": "RUNNING"
    },
    "links": {
      "self": "https://myhost/polarion/rest/v1/jobs/
8dd9cae6-927aec93-66855dab-098a0106",
      "log": "https://myhost/polarion/job-report?
jobId=8dd9cae6-927aec93-66855dab-098a0106"
    }
  }
}
```

Example

- **Example Request (import job state and status):**

  ```
  GET https://myhost/polarion/rest/v1/jobs/
  8dd9cae6-927aec93-66855dab-098a0106
  ```

- **Response Body (200 OK):**

```
{
  "links": {
    "self": "https://myhost/polarion/rest/v1/jobs/
8dd9cae6-927aec93-66855dab-098a0106"
  },
  "data": {
    "type": "jobs",
    "id": "8dd9cae6-927aec93-66855dab-098a0106",
    "attributes": {
      "jobId": "8dd9cae6-927aec93-66855dab-098a0106",
      "name": "Import of test results",
      "state": "FINISHED",
      "status": {
        "type": "OK"
      }
    },
    "links": {
      "self": "https://myhost/polarion/rest/v1/jobs/
8dd9cae6-927aec93-66855dab-098a0106",
      "log": "https://myhost/polarion/job-report?
jobId=8dd9cae6-927aec93-66855dab-098a0106"
    }
  }
}
```

Export Tests to Excel

You can export Test Results using the `exportTestsToExcel` endpoint. You can control the output of the export by supplying the following parameters in the Request Body:

- `query` (a Lucene query to select Test Cases to export)
- `sortby` (the sort criteria for the content of the export)
- `template` (the template to use for the export)

Example

- **Example Request (beginning of export):**

  ```
  POST https://myhost/polarion/rest/v1/projects/myProject/testruns/
  myTestRun/actions/exportTestsToExcel
  ```

- **Response Body (202 Accepted):**

```
{
  "data": {
    "type": "jobs",
    "id": "8e496cc1-927aec93-7f1efe40-eed209a9",
    "attributes": {
      "jobId": "8e496cc1-927aec93-7f1efe40-eed209a9",
      "name": "Export (xlsx: Microsoft Excel Tests)",
      "state": "RUNNING"
    },
    "links": {
      "self": "https://myhost/polarion/rest/v1/jobs/
8e496cc1-927aec93-7f1efe40-eed209a9",
      "log": "https://myhost/polarion/job-report?
jobId=8e496cc1-927aec93-7f1efe40-eed209a9"
    }
  }
}
```

Example

- **Example Request (export job state and status):**

  `GET https://myhost/polarion/rest/v1/jobs/8e496cc1-927aec93-7f1efe40-eed209a9`

- **Response Body (200 OK):**

```
{
  "links": {
    "self": "https://myhost/polarion/rest/v1/jobs/
8e496cc1-927aec93-7f1efe40-eed209a9"
  },
  "data": {
    "type": "jobs",
    "id": "8e496cc1-927aec93-7f1efe40-eed209a9",
    "attributes": {
      "jobId": "8e496cc1-927aec93-7f1efe40-eed209a9",
      "name": "Export (xlsx: Microsoft Excel Tests)",
      "state": "FINISHED",
      "status": {
        "type": "OK"
      }
    },
    "links": {
      "self": "https://myhost/polarion/rest/v1/jobs/
8e496cc1-927aec93-7f1efe40-eed209a9",
      "log": "https://myhost/polarion/job-report?
jobId=8e496cc1-927aec93-7f1efe40-eed209a9",
      "downloads": [
        "https://myhost/polarion/rest/v1/jobs/
8e496cc1-927aec93-7f1efe40-eed209a9/filename/tests.xlsx"
```

```
            ]
        }
    }
}
```

Example

- **Example Request (to download the exported content):**

  `GET https://myhost/polarion/rest/v1/jobs/8e496cc1-927aec93-7f1efe40-eed209a9/actions/download/tests.xlsx`

Page parameter values

Polarion **Info Pages** and **LiveReport Pages** can be accessed through the Polarion REST API to read and update their properties and default parameter values.

(Accessing or modifying the page content is not possible. Working with **Page** attachments is possible, see Attachments.)

Get Page parameters

Page parameters show among fields as special attributes with the `parameter_` prefix, but only if the particular page parameter has a non-empty default value set and it was explicitly requested to be fetched, or `fields[pages]=@all` was used.

Example

- **Example Request:**

  `GET https://myhost/polarion/rest/v1/projects/myProject/spaces/_default/pages/MyInfoPage?fields[pages]=@basic,parameter_severity`

- **Response Body:**

```
{
    "links": {

"self": "https://myhost/polarion/rest/v1/projects/myProject/spaces/
_default/pages/MyPage?fields%5Bpages%5D=%40basic%2Cparameter_severity"
    },
    "data": {
        "type": "pages",
        "id": "myProject/_default/MyPage",
        "attributes": {
            "pageName": "MyPage",
            "spaceId": "_default",
            "title": "My Report Page",
            "parameter_severity": "basic"
        },
        "relationships": {...},
        "links": {...}
    }
}
```

Update Page parameters

The following example shows the parameter with `severity` ID (presented as the `parameter_severity` attribute) and its current default value. This default value can be modified via the REST API using a PATCH request to the **Page** resource. It can be used to influence the **LiveReport** or **Info Page** content if the parameter value is used by the **Page's** widgets or scripts.

Example

- **Example Request:**

  ```
  PATCH https://myhost/polarion/rest/v1/projects/myProject/spaces/_default/
  pages/MyPage
  ```

- **Request Body:**

```
{
  "data": {
    "type": "pages",
    "id": "myProject/_default/MyPage",
    "attributes": {
      "parameter_severity": "major"
    }
  }
}
```

Collections

Collections overview

Collections support parallel development activities, help with audit and regulatory compliance and support advanced reuse scenarios.

Polarion REST API provides ways to read, create, and manipulate **Collections**.

Creating Collections

The following is an example of a request to create a new Collection in the `myProject` **Project**:

Example

- **Example Request:**

  `POST /polarion/rest/v1/projects/myProject/collections`

- **Request Body:**

```
{
  "data": [
    {
      "type": "collections",
      "attributes": {
        "description": {
          "type": "text/html",
          "value": "My text value"
        },
        "name": "Test"
      },
      "relationships": {
        "documents": {
          "data": [
            {
              "type": "documents",
              "id": "myProject/Specification/Administration
Specification",
              "revision": "1"
            }
          ]
        },
        "upstreamCollections": {
          "data": [
            {
              "type": "collections",
              "id": "myProject/20",
              "revision": "611"
            }
          ]
        }
      }
    }
  ]
}
```

- **Response Body:**

```
{
  "data": [
    {
      "type": "collections",
      "id": "elibrary/22",
      "links": {
        "self": "http://[Your_Polarion_server]/polarion/rest/v1/
projects/elibrary/collections/22",
        "portal": "http://[Your_Polarion_server]/polarion/redirect/
project/elibrary/collection?id=22"
      }
    }
  ]
}
```

- **Status Code:**

  `201`

Create Relationships for a Collection

The following is an example of a request to create specific **Relationships** for a **Collection**.

You can only post **Upstream Collections** and **Documents**.

(`author`, `project`, and `reusedFrom` are read-only **Relationships**.)

Create a Collection with a Document Relationship

We can create a **Document Relationship** inside a **Collection** using the endpoint below:

Example

- **Example Request:**

  ```
  POST /polarion/rest/v1/projects/myProject/collections/{collectionId}/
  relationships/{relationshipId}
  ```

- **Request Body:**

```
{
  "data": [
    {
      "type": "documents",
      "id": "elibrary/Specification/Administration Specification"
    }
  ]
}
```

- **Response:**

```
204 No Content
```

Create a Collection with an Upstream Collection Relationship

We can create an **Upstream Collection Relationship** inside a **Collection** using the endpoint below:

Example

- **Example Request:**

  `POST /polarion/rest/v1/projects/myProject/collections/{collectionId}/relationships/{relationshipId}`

- **Request Body:**

```
{
  "data": [
    {
      "type": "collections",
      "id": "myProject/20",
      "revision": "611"
    }
  ]
}
```

- **Response:**

  `204 No Content`

Update Collection content

You can use PATCH to update a **Collection's** content.

Example

- **Example Request:**

  `PATCH https://hostname/polarion/rest/v1/projects/myproject/collections/{collectionId}`

- **Request Body:**

```
{
  "data": {
    "type": "collections",
    "id": "MyProjectId/MyCollectionsId",
    "attributes": {
      "closedOn": "1970-01-01T00:00:00Z",
      "description": {
        "type": "text/html",
        "value": "My text value"
      },
      "name": "Name"
    },
    "relationships": {
      "elements": {
        "data": [
          {
            "type": "documents",
            "id": "MyProjectId/MySpaceId/MyDocumentId"
          }
        ]
      }
    }
  }
}
```

- **Response:**

  `204 No Content`

Update Relationship content for a Collection

The following is an example of a **Request** to update specific **Relationships** for a **Collection**.

You can only PATCH **Upstream Collections** and **Documents**.

(`author`, `project`, and `reusedFrom` are read-only **Relationships**.)

Update a Collection with a Document Relationship

You can update a **Document** Relationship inside the **Collection** using the endpoint below:

Example

- **Example Request:**

  `PATCH /polarion/rest/v1/projects/myProject/collections/{collectionId}/relationships/{relationshipId}`

- **Request Body:**

```
{
  "data": [
    {
      "type": "documents",
      "id": "elibrary/Specification/Administration Specification"
    }
  ]
}
```

- **Response:**

  `204 No Content`

Update a Collection with an Upstream Collection Relationship

You can update an **Upstream Collection Relationship** inside the **Collection** using the endpoint below:

Example

- **Example Request:**

  ```
  PATCH /polarion/rest/v1/projects/myProject/collections/{collectionId}/
  relationships/{relationshipId}
  ```

- **Request Body:**

```
{
  "data": [
    {
      "type": "collections",
      "id": "myProject/20",
      "revision": "611"
    }
  ]
}
```

- **Response:**

  `204 No Content`

Get Collections

You can use this endpoint to get all **Collections** inside the `myproject` **Project**.

Example

- **Example Request:**

  `GET /polarion/rest/v1/projects/myProject/collections`

- **Response**

  Returns a list of all the **Collections** in `myproject`.

- **Status Code:**

  `200`

Get a specific Collection

You can use this endpoint to get a specific **Collection** inside the `myproject` **Project**.

You must pass the `projectId` and `collectionId` as parameters.

Example

- **Example Request:**

  ```
  GET /polarion/rest/v1/projects/myProject/collections/{collectionId}
  ```

- **Response**

  Returns a specific **Collection** with the given `collectionId` if it is inside the **Project** with a `200` status code.

  Otherwise, it will throw a `Not Found` error with a `404` status code.

Get all Relationships of a Collection

You can get the following **Relationships** of a **Collection**:

- `documents`: To get all **Documents** inside the **Collection**.
- `upstreamCollection`: To get related **Upstream Collections**.
- `reusedFrom` If the **Collection** is reused from another **Collection**, then those details are returned.
- `author`: Returns the **Collection** author's details.
- `project`: Returns the **Project** details.

You must pass the `projectId` and `collectionId`, and the **Relationship IDs** will follow. You can access a single **Relationship** at a time.

Example

- **Example Request:**

  ```
  GET /polarion/rest/v1/projects/myProject/collections/{collectionId}/
  relationships/{relationshipId}
  ```

- **Status Code:**

```
200
```

Delete all Collections

You can use this endpoint to delete all **Collections** inside the `myproject` **Project**.

Example

- **Example Request:**

```
DELETE /polarion/rest/v1/projects/myProject/collections
```

- **Response:**

```
204 No Content
```

Delete a specific Collection

You can use this endpoint to delete a specific **Collection** inside the `myproject` **Project**.

You must pass `projectId` and `collectionId` as parameters.

Example

- **Example Request:**

```
DELETE /polarion/rest/v1/projects/myProject/collections/{collectionId}
```

- **Response:**

```
204 No Content
```

Delete a specific Relationship from a Collection

You can delete **Relationships** of a **Collection**.

- `documents`: To get all **Documents** inside the **Collection**.
- `upstreamCollection`: To get related **Upstream Collections**.
- `reusedFrom`: A read-only **Relationship**. Cannot delete.
- `author`: A read-only **Relationship**. Cannot delete.
- `project`: A read-only **Relationship**. Cannot delete.

You must pass the `projectId` and `collectionId`. The **Relationship IDs** will follow.

You can access a single **Relationship** at a time.

Note

- **Example Request:**

  ```
  DELETE /polarion/rest/v1/projects/myProject//collections/{collectionId}/
  relationships/{relationshipId}
  ```

Delete a Document Relationship from a Collection

You can delete **Document Relationships** from a **Collection** with the following endpoint.

Example

- **Example Request:**

  ```
  DELETE /polarion/rest/v1/projects/myProject/collections/{collectionId}/
  relationships/{relationshipId}
  ```

- **Request Body:**

  ```
  {
    "data": [
      {
        "type": "documents",
        "id": "elibrary/Specification/Administration Specification"
      }
    ]
  }
  ```

- **Response:**

  ```
  204 No Content
  ```

Delete an Upstream Collection relationship from a Collection

You can delete an **Upstream Collection Relationship** from a **Collection** with the endpoint below:

Example

- **Example Request:**

  ```
  DELETE /polarion/rest/v1/projects/myProject/collections/{collectionId}/
  relationships/{relationshipId}
  ```

- **Request Body:**

```
{
  "data": [
    {
      "type": "collections",
      "id": "myProject/20",
      "revision": "611"
    }
  ]
}
```

- **Response:**

  `204 -No content`

Close a Collection

You can **Close** a **Collection** with the endpoint below:

Example

- **Example Request:**

  `POST /polarion/rest/v1/projects/myProject/collections/{collectionId}/actions/close`

  Add the `projectId` and `collectionId` as parameters.

- **Response:**

  Collection is closed

  `204-No content`

Note

- If you execute the endpoint for a **Collection** that is already closed, it returns `204 NO CONTENT`.
- All **Documents** inside the **Collection** should have a revision other than `HEAD` otherwise, a `400 BAD REQUEST` error is thrown, and the following message appears:

  `Change all HEAD Documents to a baseline or revision before closing the Collection.`

- Any attempt to change a closed **Collection** (except reopening) returns `400` with the error message.

Reopen a Collection

You can **Reopen** a **Collection** with the endpoint below:

Example

- **Example Request:**

  ```
  POST /polarion/rest/v1/projects/myProject/collections/{collectionId}/
  actions/reopen
  ```

  Add the `projectId` and `collectionId` as parameters.

- **Response:**

  Collection is reopened.

  ```
  204-No content
  ```

Note

Hitting the endpoint for an already opened **Collection** returns `204 NO CONTENT`.