Structure Resource

This page describes resources with which you can list, create, read, update, and delete structures. Structures contain general information such as name and permissions, but not the hierarchy itself. Issue hierarchy is accessed through the Forest Resource. This page also documents structure shape and its fields, and the error entity that may be returned in case of the REST API user error.

Structure resource belongs to **version 2.0** of the API.

`/structure/ GET`	list structures
`/structure/ POST`	create a structure
`/structure/{id} GET`	read structure
`/structure/{id}/update POST`	update one or several structure fields
`/structure/{id} DELETE`	delete structure

Quick navigation:

Structure Representations

Structure is represented via JSON. All resources are also capable of producing XML.

```
{
  "id": 103,
  "name": "Structure with all fields",
  "description": "Voilà! This structure exhibits all fields.",
  "readOnly": "true",
  "editRequiresParentIssuePermission": true,
  "permissions": [
    {
      "rule": "apply",
      "structureId": 102
    },
    {
      "rule": "set",
      "subject": "group",
      "groupId": "jira-developers",
      "level": "edit"
    },
    {
      "rule": "set",
      "subject": "projectRole",
      "projectId": 10010,
      "roleId": 10020,
      "level": "admin"
    },
    {
      "rule": "set",
      "subject": "anyone",
      "level": "view"
    },
    {
      "rule": "set",
      "subject": "user",
      "username": "agentk",
      "level": "none"
    }
  ],
  "owner": "user:admin"
}
```

Top

Structure Fields

Structure objects accessible through these resources have the following fields, most of which represent structure details as outlined in the Structure User's Guide:

`id`	The ID of the structure (integer, `1..2^63 - 1`.)
`name`	The name of the structure. A structure must have a non-empty name, which does not have to be unique.
`description`	The description on the structure. May be absent.
`readOnly`	`true` if the user has only View access level to the structure, otherwise absent.
`editRequiresParentIssuePermission`	`true` if the Require Edit Issue Permission on Parent Issue flag is set on this structure, otherwise absent.
`permissions`	The list of structure permission rules. Present only if the user has Control access level to the structure. Some resources do not include permissions unless requested to do so. List order is as important as the rules themselves.
`owner`	The owner of the structure. Present only if the user is the owner of this structure or if he has Browse Users permission. A string of the form `user:USERNAME`, where USERNAME is the JIRA user login name. Example: `user:jsmith`.

Please note that structure resources described on this page do not include information about issue hierarchies. The content of a structure, i.e. its hierarchy of items, can be read or modified using Forest Resource.

Top

Permission rules

There are two types of permission rules, those that *set* permissions and those that *apply* permissions from another structure. They have different fields depending on the type.

Set rules

`rule`	Must be equal to `set`, case-insensitive.
`subject`	Identifies the type of the subject to which the rule applies. Must be one of `group, projectRole, user, anyone`. See below how to identify the subject.
`level`	Access level to set to the specified subject. Must be equal to one of the names of the Permission Level enum constants, case-insensitive. Please note that Control permission is represented by the ADMIN enumeration constant.

In addition, there are fields to identify the subject.

`group`

The rule applies to all users within the JIRA group.

`groupId`	The name of the JIRA group. Example: `jira-developers`.

REST API user can create such rule only for a group he belongs to.

`projectRole`

The rule applies to all users that have a role in a project.

`projectId`	The ID of the project. Example: `10010`.
`roleId`	The ID of the role. Example: `10010`.

REST API user can create such rule only for roles in projects where Structure is enabled, and for which he has Browse Projects permission.

`user`

The rule applies to the user.

`username`	Name of the user. Example: `jsmith` for user John Smith.

REST API user can create such rule only if he has Browse Users permission, and if such user exists.

anyone

The rule applies to all users, even anonymous (not authenticated.) The rule shouldn't have any additional fields.

Apply rules

`rule`	Must be equal to `apply`, case-insensitive.
`structureId`	The ID of the structure which permissions should be applied. Example: `112`

Apply rule creates a dependency on another structure. Circular dependencies are not allowed. Also, a REST API user can create such rule only if he has Control access level to the referenced structure.

Top

Error entity

```
{
  "code": 4005,
  "error": "STRUCTURE_NOT_EXISTS_OR_NOT_ACCESSIBLE[4005]",
  "structureId": 160,
  "message": "Referenced structure [160] does not exist or you don't have Control permissions on it.",
  "localizedMessage": "Das Struktur [160] existiert nicht oder sie haben keine Kontrolle Berechtigungen."
}
```

In some cases, requests to structure resources result in an error response containing an error entity. Any of its fields may be absent.

`code`	Integer code of the error
`error`	Brief technical description of the error. Contains a name of the corresponding StructureError enum constant.
`structureId`	The ID of the structure involved.
`issueId`	The ID of the JIRA issue involved.
`message`	More detailed message, may contain technical details.
`localizedMessage`	User-displayable message in the REST API user locale or JIRA default locale if the user is not authenticated.

Top

Structure Resources

GET /structure

```
GET $baseUrl/rest/structure/2.0/structure
GET $baseUrl/rest/structure/2.0/structure?
name=$name&permission=$permission&withPermissions=$withPermissions&withOwner=$withOwner&limit=100
```

A list of all structures visible to the REST API user. Optionally, the result can be filtered by name or user's access level. By default, permission rules and owners are not included, you should use query parameters if you want them to be included.

> **Who can access this resource**
>
> All users who have access to the Structure Plugin. The returned list contains only structures to which the REST API user has at least View access level.

Request

Query parameters:

`name`	If present, the returned list will contain only structures which names contain the specified string (case insensitive).
`permission`	If present, the returned list will contain only structures to which the REST API user has the specified access level. Must be equal to one of the names of the Permission Level enum constants, case-insensitive. NONE is treated in the same way as VIEW. Please note that Control permission is represented by the ADMIN enumeration constant.
`withPermissions`	If `true`, permission rules will be included in the response. Default is `false`.
`withOwner`	If `true`, owner will be included in the response. Default is `false`.
`archived`	If `true`, the returned list can also contain archived structures. Default is `false`.
`limit`	If specified, must be a number. Defines the maximum number of structures to return.

Each of the filter parameters `name`, `permission`, or `issueId` can be specified only once, otherwise the first is used. Different parameters are combined with AND.

HTTP headers:

`Content-Type`	Should be one of `application/json`, `application/xml`.
`Accept`	Should be one of `application/json`, `application/xml`.

Response

Success

`200 OK`	Response entity contains the only field, `structures`, which contains the list of the structure objects, sorted by name.	`application/json`, `application/xml`

Example 1: all structures

```
GET $baseUrl/rest/structure/2.0/structure
```

```
{
  "structures": [
    {
      "id": 1,
      "name": "Global Structure",
      "description": "Initial general-purpose structure.",
      "editRequiresParentIssuePermission": true
    },
    {
      "id": 102,
      "name": "Test plan",
      "description": "Test plan #3",
      "readOnly": true
    },
    {
      "id": 100,
      "name": "Test plan",
      "description": "Test plan #1"
    },
    {
      "id": 101,
      "name": "Test plan",
      "description": "Test plan #2"
    }
  ]
}
```

Example 2: only "Test plan"

```
GET $baseUrl/rest/structure/2.0/structure?name=test+plan
```


```
{
  "structures": [
    {
      "id": 102,
      "name": "Test plan",
      "description": "Test plan #3",
      "readOnly": true
    },
    {
      "id": 100,
      "name": "Test plan",
      "description": "Test plan #1"
    },
    {
      "id": 101,
      "name": "Test plan",
      "description": "Test plan #2"
    }
  ]
}
```


Example 3: structures that the user can edit with permissions and owners shown

```
GET $baseUrl/rest/structure/1.0/structure?permission=edit&withPermissions=true&withOwner=true
```

```
{
  "structures": [
    {
      "id": 1,
      "name": "Global Structure",
      "description": "Initial general-purpose structure.",
      "editRequiresParentIssuePermission": true
    },
    {
      "id": 100,
      "name": "Test plan",
      "description": "Test plan #1",
      "permissions": [
        {
          "rule": "set",
          "subject": "group",
          "groupId": "jira-users",
          "level": "edit"
        },
        {
          "rule": "set",
          "subject": "projectRole",
          "projectId": 10010,
          "roleId": 10010,
          "level": "none"
        },
        {
          "rule": "apply",
          "structureId": 101
        }
      ],
      "owner": "user:jsmith"
    },
    {
      "id": 101,
      "name": "Test plan",
      "description": "Test plan #2",
      "owner": "user:admin"
    }
  ]
}
```


Example 4: require XML representation

Note that the same can be achieved by specifying `application/xml` in the `Accept` HTTP header.

```
GET $baseUrl/rest/structure/1.0/structure.xml?name=test+plan
```


```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<structureList>
  <structures>
    <structure>
      <id>100</id>
      <name>Test plan</name>
      <description>Test plan #1</description>
    </structure>
  </structures>
</structureList>
```


Error

`400 Bad Request`	`permission` parameter is set to an unknown value, or request is invalid for other reasons. In the first case, response contains error entity, in the second it is empty.	`application/json`, `application/xml`

403 Forbidden	If Structure Plugin is not accessible to the REST API user, or if issue with ID `issueId` does not exist or the REST API user does not have enough permissions to access it. Response contains error entity.	`application/json`, `application/xml`
404 Not Found	If `issueId` is not an integer. Response entity contains a standard JIRA error HTML page.	`text/html`
500 Internal Server Error	If an internal error has occurred while processing this request.	
503 Service Unavailable	If Structure Plugin is stopped at the time of request. For example, the Restore operation may be in progress.	

Other return codes are possible under the normal rules of HTTP communication.

Top

POST /structure

```
POST $baseUrl/rest/structure/2.0/structure
```

Create an empty structure by POSTing to this resource.

> **Who can access this resource**
>
> Only logged in users who have access to the Structure Plugin and a permission to create structures.

Request

Request entity should contain the new structure. Structure name, `name`, must be present and non-empty. Fields `id`, `readOnly`, and `owner` are ignored. All rules in `permissions` are validated according to their respective rule types.

Please note that this resource accepts only JSON structure representation.

HTTP headers:

`Content-Type`	Must be `application/json`.
`Accept`	Should be one of `application/json`, `application/xml`.

Response

Success

201 Created	Response entity contains the created structure with fields, including `permissions` and `owner`.	`application/json`, `application/xml`

Example 1: minimal structure

```
POST $baseUrl/rest/structure/2.0/structure
```

Request entity	Response entity
`{ "name":"Test plan" }`	`{ "id": 104, "name": "Test plan", "description": "", "permissions": [], "owner": "user:admin" }`

Example 2: structure with some permissions

```
POST $baseUrl/rest/structure/2.0/structure
```

Request entity	Response entity
`{ "name":"Structure with some permissions", "editRequiresParentIssuePermission":"true", "permissions":[ { "rule":"apply", "structureId":102 } ] }`	`{ "id": 105, "name": "Structure with some permissions", "description": "", "editRequiresParentIssuePermission": true, "permissions": [ { "rule": "apply", "structureId": 102 } ], "owner": "user:admin" }`

Error

`400 Bad Request`	Structure data is not well-formed (syntax error) or invalid (semantic error.) Not well-formed structure data examples: request JSON is syntactically incorrect; JSON contains unknown field; `name` is not present or empty; `permissions` list contains a *set* rule with `level` set to an invalid value. Invalid structure example: `permissions` list contains a rule that fails validation. Response entity contains error. Problems with *apply* rule usually have `structureId` to indicate the invalid reference.	`application/json`, `application/xml`
`403 Forbidden`	If REST API user is not logged in or does not have permissions to access Structure Plugin or to create structures. Response contains error entity.	`application/json`, `application/xml`
`500 Internal Server Error`	If an internal error has occurred while processing this request.	
`503 Service Unavailable`	If Structure Plugin is stopped at the time of request. For example, the Restore operation may be in progress.	

Other return codes are possible under the normal rules of HTTP communication.

Top

GET /structure/{id}

```
GET $baseUrl/rest/structure/2.0/structure/$id
GET $baseUrl/rest/structure/2.0/structure/$id?withPermissions=$withPermissions&withOwner=$withOwner
```

This resource allows to obtain structure details for the particular structure. By default, `permissions` and `owner` are not included, use query parameters to include them.

> **Who can access this resource**
>
> All users who have access to the Structure Plugin. To access the particular structure, the user has to have at least View access level.

Request

Path parameter:

`id`	the ID of the structure

Query parameters:

`withPermissions`	If `true`, permission rules will be included in the response. Default is `false`.
`withOwner`	If `true`, owner will be included in the response. Default is `false`.

HTTP headers:

`Content-Type`	Should be one of `application/json`, `application/xml`.
`Accept`	Should be one of `application/json`, `application/xml`.

Response

Success

`200 OK`	Response entity contains the created structure along with all of its fields. Field `permissions` is included if the REST API user has Control permission on this structure. Field `owner` is included if the REST API user is either the owner of this structure or has Browse Users permission.	`application/json`, `application/xml`

Example 1: retrieve structure with ID 100 without permissions and owner

```
GET $baseUrl/rest/structure/2.0/structure/100
```

```
{
  "id": 100,
  "name": "Test plan",
  "description": "Test plan #1"
}
```

Example 2: permissions and owner are requested to be included, but only owner is shown, because the user has only View access as indicated by `readOnly`

```
GET $baseUrl/rest/structure/2.0/structure/102?withOwner=true&withPermissions=true
```

```
{
  "id":102,
  "name":"Test plan",
  "description":"Test plan #3",
  "readOnly":true,
  "owner":"user:admin"
}
```

Example 3: XML representation may be requested in the request URL instead of the `Content-Type` HTTP header

```
GET $baseUrl/rest/structure/2.0/structure/102.xml
```

```
<structure>
  <id>102</id>
  <name>Test plan</name>
  <description>Test plan #3</description>
  <readOnly>true</readOnly>
</structure>
```

Error

`400 Bad Request`	One of the query parameters is too long.	

`403 Forbidden`	If REST API user does not have permissions to access Structure Plugin or does not have at least View permission on this structure. Response contains error entity.	`application/json`, `application/xml`
`404 Not Found`	If `id` is not an integer in `1..2^63-1`. Response entity contains a standard JIRA error HTML page.	`text/html`
`500 Internal Server Error`	If an internal error has occurred while processing this request.	
`503 Service Unavailable`	If Structure Plugin is stopped at the time of request. For example, the Restore operation may be in progress.	

Other return codes are possible under the normal rules of HTTP communication.

Top

POST /structure/{id}/update

```
POST $baseUrl/rest/structure/1.0/structure/$id/update
```

Update one or several fields of a structure by POSTing to this resource.

> **Who can access this resource**
>
> Only logged in users who have access to the Structure Plugin and Control permission on this structure.

Request

Request entity should contain those structure fields that need to be changed. Non-present fields will not be changed (for this user; `readOnly` may change for other users as a result of changing `permissions`.) Fields `id`, `readOnly`, and `owner` are ignored.

Please note that `permissions` field is modified as a whole, so to add a rule, you have to provide the new list of rules in the proper order.

If `permissions` field is present, all rules are validated according to their respective rule types.

Please note that this resource accepts only JSON structure representation.

HTTP headers:

`Content-Type`	Must be `application/json`.
`Accept`	Should be one of `application/json`, `application/xml`.

Response

Success

`200 OK`	Response entity contains the updated structure with all fields, including `permissions` and `owner`.	`application/json`, `application/xml`

Example 1: change description of the Global Structure

```
POST $baseUrl/rest/structure/1.0/structure/1/update
```

Request entity	Response entity

`{ "description":"Company-wide structure providing the Big Picture." }`	`{ "id":1, "name":"Global Structure", "description":"Company-wide structure providing the Big Picture.", "editRequiresParentIssuePermission":true, "permissions":[{ "rule":"set", "subject":"anyone", "level":"view" }, { "rule":"set", "subject":"group", "groupId":"jira-users", "level":"edit" }, { "rule":"set", "subject":"group", "groupId":"jira-administrators", "level":"admin" }] }`

Example 2: changing permission rules

```
POST $baseUrl/rest/structure/1.0/structure
```

Request entity	Response entity
`{ "permissions":[ { "rule":"set", "subject":"group", "groupId":"jira-users", "level":"edit" }, { "rule":"apply", "structureId":101 } ] }`	`{ "id": 105, "name": "Structure with some permissions", "description": "", "editRequiresParentIssuePermission": true, "permissions": [ { "rule": "set", "subject": "group", "groupId": "jira-users", "level": "edit" }, { "rule": "apply", "structureId": 101 } ], "owner": "user:admin" }`

Error

`400 Bad Request`	Structure data is not well-formed (syntax error) or invalid (semantic error.) Not well-formed structure data examples: request JSON is syntactically incorrect; JSON contains unknown field; `permissions` list contains a *set* rule with `level` set to an invalid value. Invalid structure example: `permissions` list contains a rule that fails validation. Response entity contains error. Responses to problems with an *apply* rule usually have `structureId` to indicate the invalid reference.	`application/json`, `application/xml`.

`403 Forbidden`	If REST API user is not logged in, does not have permissions to access Structure Plugin, or does not have Control access level to this structure. Response contains error entity.	`application/json`, `application/xml`
`500 Internal Server Error`	If an internal error has occurred while processing this request.	
`503 Service Unavailable`	If Structure Plugin is stopped at the time of request. For example, the Restore operation may be in progress.	

Other return codes are possible under the normal rules of HTTP communication.

Top

DELETE /structure/{id}

Deletes the designated structure.

> **Who can access this resource**
>
> Only logged in users who have access to the Structure Plugin and Control permission on this structure.

Request

Path parameter:

`id`	the ID of the structure

HTTP headers:

`Content-Type`	Must be `application/json`.
`Accept`	Should be absent or equal to one of `application/json`, `application/xml`.

Response

Success

`200 OK`	Contains an object with the only field `empty` with value `true`.	`application/json`, `application/xml`

Note: it should have been `204 No content` instead, but there were reports of some browsers (Firefox) incorrectly processing such results, so it's as it is.

Example

```
DELETE $baseUrl/rest/structure/1.0/structure/108
```

```
{
  "empty": true
}
```

Error

`403 Forbidden`	If REST API user is not logged in, does not have permissions to access Structure Plugin, or does not have Control access level to this structure. Response contains error entity.	`application/json`, `application/xml`
`404 Not Found`	If `id` is not an integer in `1..2^63-1`. Response entity contains a standard JIRA error HTML page.	`text/html`
`404 Not Found`	If `id` is an integer in `1..2^63-1`, but the structure with the specified `id` does not exist or the user does not have View access level to it.	`application/json`, `application/xml`
`500 Internal Server Error`	If an internal error has occurred while processing this request.	
`503 Service Unavailable`	If Structure Plugin is stopped at the time of request. For example, the Restore operation may be in progress.	

Other return codes are possible under the normal rules of HTTP communication.

Top