How to Use a Store(Issue) Function

Last Modified on 04/15/2024 4:29 am EDT

This article describes how to use a *store(issue)* function that is available starting from versions 5.0.87.1 and 5.1.4 and up. The *store(issue)* function can be added to the Incoming script rules for multiple purposes. It allows performing multiple consecutive operations within one synchronization. Here is an example of how to use the *store(issue)* function to define the order of the operations that are being performed.

Incoming script rules

```
if (firstSync) {
  issue.projectKey = "FOO"
  issue.typeName = "Task"
  issue.summary = "Hello there"
  store(issue) // creates the issue with only project, type and summary
}
issue.setStatus("In Progress")
store(issue) // transitions the issue to "In Progress"
issue.comments = commentHelper.mergeComments(issue, replica) // adds comments after changing the status of th
e issue
```

Another example of multiple operations:

Incoming script rules

```
issue.comments = commentHelper.addComment("Let's start progress", issue.comments)
issue.setStatus("In Progress")
store(issue)

issue.fixVersions = nodeHelper.getVersion("1.0.0", issue.project)
issue.setStatus("In Review")
store(issue)

issue.resolution = nodeHelper.getResolution("Fixed")
issue.setStatus("Resolved")
store(issue)
```

Sometimes workflow configuration in Jira does not let you make changes under certain circumstances, like, for example, when the issue is closed. To make changes to the issue in this case it is required to open the issue first. This is when the *store(issue*) function becomes a handy solution. Exalate will check the issue status and will act accordingly. Depending on the issue status the ordering of the actions will differ.

Incoming script rules

```
if (replica.status.name == "Closed" && issue.status.name != "Closed") {
  issue.description = replica.description
  store(issue) // first update the issue
  issue.setStatus("Closed") // then close it, because you won't be able to modify closed issues
}

else if (replica.status.name != "Closed" && issue.status.name == "Closed") {
  issue.setStatus(replica.status.name)
  store(issue) // first reopen the issue
  issue.description = replica.description // then update it, because you won't be able to modify issues which are still cl
osed
}
```

Another valuable outcome is that the *store(issue*) function creates an issue ID that can be useful when creating REST API requests using this issue ID. If the *store(issue*) is not present in the scripting rules, the issue ID won't get created to be used in REST API calls. Otherwise, by adding the *store(issue*) function in the script you can customize the scripting rules further by using the issue ID.

Incoming script rules

```
issue.summary = "Summary"
issue.typeName = "Story"
issue.projectKey = "PROJA"
store(issue)// creates an issue ID
def issueId = issue.id // now it's guaranteed not to be empty
def votes = new JiraClient(httpClient).http("GET", "/rest/api/3/issue/${issueId}/votes", [:], null, [:]) { response ->
    if (response.code >= 300 && response.code != 404) {
        throw new com.exalate.api.exception.IssueTrackerException("Failed to perform the request GET  /rest/api/3/issu
e/${issue.id}/votes (status ${response.code}), and body was: \n\"response.body\"\nPlease contact Exalate Support: ".
toString() + response.body())
    }
    if (response.code == 404) {
        return null
    }
    def responseStr = response.body as String
    responseStr ?
        new groovy.json.JsonSlurper().parseText(responseStr) as Map<String, Object>
        : null;
}?.votes
```


Product

About Us

Release History

Glossary

API Reference

Security

Pricing and Licensing

Resources

Subscribe for a weekly Exalate hack

Academy

Blog

YouTube Channel

Ebooks

Still need help?

Join our Community

Visit our Service Desk

Find a Partner