How to Sync Comments in Azure DevOps Server & Service

Last Modified on 01/15/2026 6:07 pm EST

This article shows how to synchronize comments.

You can handle comments in different ways:

- merge local and received comments (this is a default behavior)
- filter comments
- format received comments

Source Side

Outgoing sync

You can decide which comments to share:

Synchronize comments that have no group or role level assign

```
replica.comments = commentHelper.filterLocal(issue.comments)
```

Service Desk: Synchronize only public comments

```
replica.comments = issue.comments.findAll{!it.internal}
```

Make sure the customer request type is assigned to the issue that you are synchronizing.

Send only comments created by a certain user

```
//send only Luke Skywalker's comments
replica.comments = issue.comments.findAll { comment -> comment.author.displayName == "Luke Skywalker" }
```

Destination Side

Incoming sync

You can perform the following actions on comments locally:

- merge local and remote comments
- add remote comments
- filter: add comments created by a specific user
- format comments using custom formatting

You can add new comments and keep the existing comments updated using mergeComments comment helper. This method would prepend the comment content with the author of the original comment. This is the default behavior.

```
issue.comments = commentHelper.mergeComments(issue, replica)
```

Apply received comments without custom formatting

The comment is added just with the body from the original comment, without the author of the original comment.

```
issue.comments = commentHelper.mergeComments(issue, replica, {it})
```

Create received comments as internal

```
issue.comments = commentHelper.mergeComments(issue, replica, {it.internal = true})
```

Add comments every time the comment on the remote side was edited

This is an example of a case, where comments editing is disabled, but you still want to get updates every time the remote comment was updated.

```
issue.comments += replica.addedComments
```

Manipulate the comments via helper methods

With the help of commentHelper methods, you can manipulate comments and define how to apply received comments on the local issue.

```
// Format each remote comment - add the author

issue.comments = commentHelper.mergeComments(issue, replica,
                    {
                      comment ->
                      comment.body =
                        "[" + comment.author.displayName +
                              "| email: " + comment.author.email + "]" +
                              " commented: \n" +
                        comment.body + "\n"
                    }
)
```

Gather statistics from comments

This is an example of groovy collection methods usage. It helps to get better control over the collection contents.

```
def numberOfCommentsPerAuthors = issue.comments.inject([:]) { result, comment ->
  def numberOfCommentsPerAuthor = result[comment.author.key]
  numberOfCommentsPerAuthor = numberOfCommentsPerAuthor ?: 0
  result[comment.author.key] = numberOfCommentsPerAuthor + 1
  result
}
```

Example for a comment list if you use the script above

```
[
        [ author:[key:"luke.skywalker"], body:"I'm a jedi knight, like my father before me" ],
        [ author:[key:"darth.vader"], body:"I am your father!" ],
        [ author:[key:"luke.skywalker"], body:"Noooooooooo!" ]
]
numberOfCommentsPerAuthors would be ["luke.skywalker" : 2,"darth.vader" : 1]
```