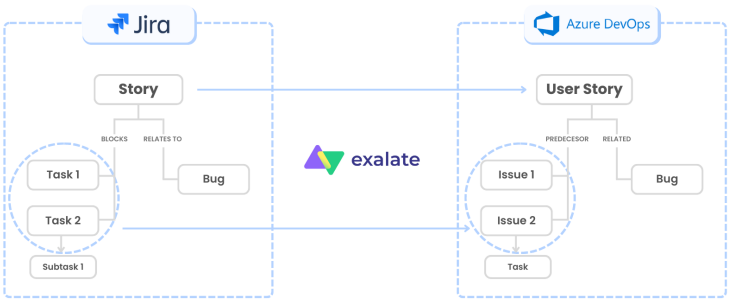
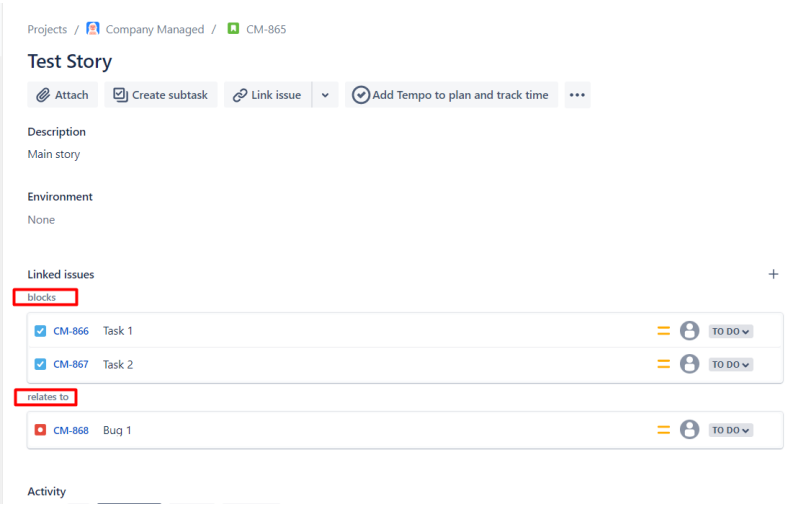
Jira Cloud Azure DevOps: Issue links hierarchy and mappings

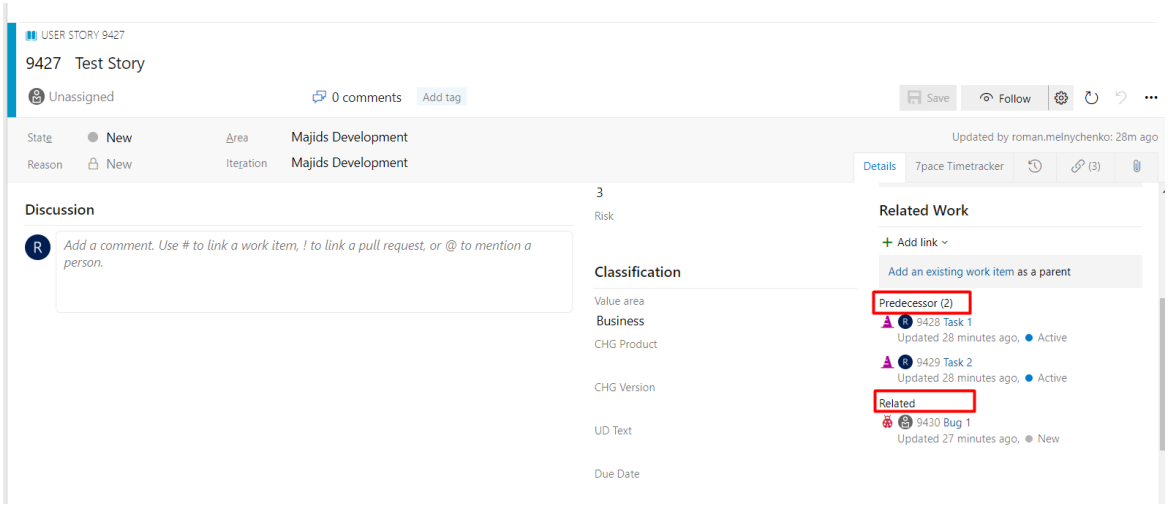
In Jira Cloud we can create issue links to other issues in Jira and define the relationship between the different issues. This use case will revolve around picking up the issueLinks from Jira Cloud and transferring them over to the Azure DevOps side using a mapping:



In this example we will map two relationships from Jira to relationships in Azure DevOps. The starting point in Jira Cloud would be as follows:



And this would end up in Azure DevOps as the following:



In order to achieve this use case, the only thing that Jira has to do is to send out the issue links, and the parentId (for sub-tasks). This is achieved by using the following lines:

Jira Cloud Outgoing

```
replica.linkedIssues = issue.issueLinks

replica.parentId       = issue.parentId
```

On the Azure DevOps side, you will need to add a parent-child relationship corresponding to the task sub-task relation in Jira. This is achieved with the following code snippet:

Azure DevOps Incoming

```
workItem.parentId = null
if (replica.parentId) {
   def localParent = syncHelper.getLocalIssueKeyFromRemoteId(replica.parentId.toLong())
   if (localParent)
      workItem.parentId = localParent.id
}
```

The next part covers the link mapping and link creation on the Azure side. This is achieved by creating a mapping first, and then calling an endpoint to get the relevant links created in Azure DevOps. This can be done using the following snippet:

Azure DevOps Incoming

```
def linkTypeMapping = [
    "blocks": "System.LinkTypes.Dependency-Reverse",
    "relates to": "System.LinkTypes.Related"
]
def linkedIssues = replica.linkedIssues
if (linkedIssues) {
    replica.linkedIssues.each{
       def localParent = syncHelper.getLocalIssueKeyFromRemoteId(it.otherIssueId.toLong())
       if (!localParent?.id) { return; }
       localUrl = baseUrl + '/_apis/wit/workItems/' + localParent.id
     def createIterationBody = [
            [
                op: "test",
                path: "/rev",
                value: (int) res.value.size()
            ],
            [
                op:"add",
                path:"/relations/-",
                value: [
                    rel:linkTypeMapping[it.linkName],
                    url:localUrl,
                    attributes: [
                        comment:""
                    ]
                ]
            ]
        ]


def createIterationBodyStr = groovy.json.JsonOutput.toJson(createIterationBody)
        converter = scala.collection.JavaConverters;
        arrForScala = [new scala.Tuple2("Content-Type","application/json-patch+json")]
        scalaSeq = converter.asScalaIteratorConverter(arrForScala.iterator()).asScala().toSeq();
        createIterationBodyStr = groovy.json.JsonOutput.toJson(createIterationBody)
        def result = await(httpClient.azureClient.ws
            .url(baseUrl+"/${project}/_apis/wit/workitems/${workItem.id}?api-version=6.0")
            .addHttpHeaders(scalaSeq)
            .withAuth(token, token, play.api.libs.ws.WSAuthScheme$BASIC$.MODULE$)
            .withBody(play.api.libs.json.Json.parse(createIterationBodyStr), play.api.libs.ws.
JsonBodyWritables$.MODULE$.writeableOf_JsValue)
            .withMethod("PATCH")
            .execute())

    }
}
```

The entire code for the Azure DevOps incoming side is attached here..

A full video demonstration of the issue can be viewed here:


Jira ADO IssueLinks mapping.mp4