Set asset custom field value with searching assets

In this example, we will define a post function to set asset custom fields value. First asset CF must be empty and we perform a custom search to locate to asset and then assign it to the asset custom field.

Let's say we have Computer asset type and a JIRA User attribute which has unique values (for this example)

When Tyler creates a service desk issue and leaves the asset custom field value empty, we'll find "Computer 01" and set it to the custom field.

Adding post function

Post function is added to Create Issue Transition
**Parameters of post function**

- Asset custom fields (with values) to inject as script parameter: Select the custom field to set. In this example "Assets - 10227" is picked
- Groovy script: Add tht groovy script and modify the parameters
  - assetTypeName: asset type name to search
  - def fieldName: attribute name to search
  - fieldValue: attribute value to search
  - AUTH_HASH: auth hash for authorization. The sample is for user=admin, and password=admin
  - assetCustomFieldId: id of the asset custom field. To find it quickly, right click on issue edit screen where asset custom field appears and find the id of it. example: = customfield_10227

**Post function groovy script (expand to see)**
import com.atlassian.jira.component.ComponentAccessor
import com.atlassian.jira.issue.CustomFieldManager
import com.atlassian.jira.issue.ModifiedValue
import com.atlassian.jira.issue.util.DefaultIssueChangeHolder
import groovy.json.JsonSlurper
import inventoryplugin.workflow.function.genericscript.dto.AssetCustomField
import inventoryplugin.workflow.function.genericscript.dto.AssetCustomFieldAndValue
import org.apache.http.util.EntityUtils
import org.apache.log4j.Logger
import org.apache.log4j.Level

/***************************************************************************************************/
/* IMPORTANT:
/* Configure these settings according to your environment
/* AUTH_HASH value must be changed according to the user who has Browse Assets for all types */
/***************************************************************************************************/
def assetTypeName = 'Computer'; // asset type name to search
def fieldName = 'JIRA User'; // attribute name to search
def fieldValue = issue.reporter.username; // attribute value to search

// IMPORTANT:
// Configure these settings according to your environment
// AUTH_HASH value must be changed according to the user who has Browse Assets for all types */
def AUTH_HASH = 'Basic YWRtaW46YWRtaW4='; // auth hash for authorization. this sample is for user=admin, and
password=admin

def assetCustomFieldId = "customfield_10227"; // id of the asset custom field. To find it quickly, right click
on issue edit screen where asset custom field appears and find the id of it.

Logger logger = Logger.getLogger("inventoryplugin.groovy.script")

/**
 * searches for assets that matches fieldName=fieldValues for the given assetTypeName and returns matching
asset ids as formatted for asset custom field (i.e: ,12,343,545,)
*/
def getAssetFieldValue(authHash, assetTypeName, fieldName, fieldValue) {
    Logger logger = Logger.getLogger("inventoryplugin.groovy.script")
    try {
        if (fieldValue) {
            // get asset ID(s)
            def baseurl = ComponentAccessor.getApplicationProperties().getString("jira.baseurl")
            // this is the search query parameters. When you search assets on Asset Navigator,
            // same parameters appear on Developer tools-> Network Tab. You can make different searches and get
            the parameters from there when you need.
            def queryToPostJson = "{"searchType" : "basic", \n            "listType" : "detail", \n            "queryIndexSearchParams" : [{\n            "field" : "" + fieldName + "", \n            "keyword" : "" + fieldValue + "", \n            "fieldType" : "STRING", \n            "range" : false\n            },\n            {\n            "field" : "form.name", \n            "keyword" : null, \n            "fieldType" : "LIST", \n            "range" : false\n            }],\n            "AUTH_HASH" : "Basic YWRtaW46YWRtaW4="
        }
    }
}
CloseableHttpClient httpclient = HttpClients.createDefault();
def httpPost = new HttpPost(baseurl + "/rest/jip-api/1.0/index/query");
httpPost.setHeader("Accept", "application/json");
httpPost.setHeader("Content-type", "application/json");
httpPost.setHeader("Authorization", authHash);
def entity = new StringEntity(queryToPostJson);
httpPost.setEntity(entity);
CloseableHttpResponse response = httpclient.execute(httpPost);
try {
  if (response.getStatusLine().getStatusCode() == 200) {
    BufferedReader br = new BufferedReader(new InputStreamReader(response.getEntity().
getContent()));
    StringBuilder sb = new StringBuilder();
    String line;
    while ((line = br.readLine()) != null) {
      sb.append(line + "\n");
    }
    br.close();
    EntityUtils.consume(response.getEntity());
    def jsonSlurper = new JsonSlurper()
    def json = jsonSlurper.parseText(sb.toString())
    def assetIdsArray = json.recordIds;
    def assetIdsSb = new StringBuffer()
    if (assetIdsArray != null && assetIdsArray.size() > 0) {
      assetIdsArray.each {
        assetIdsSb.append(',').append(it)
      }
      assetIdsSb.append(',
    }
    return assetIdsSb.toString()
  } else {
    return null;
  }
} finally {
  httpclient.close();
}

/**
 * updates asset custom field of the given issue
 */
def updateAssetCustomFieldOfTheIssue(customfieldId, issueKey, assetCustomFieldValue) {
  def issueToUpdate = ComponentAccessor.getIssueManager().getIssueObject(issueKey);
  CustomFieldManager customFieldManager = ComponentAccessor.getCustomFieldManager()
  def assetCf = customFieldManager.getCustomFieldObject(customfieldId) // asset cf id
  // execute update
  assetCf.updateValue(null, issueToUpdate, new ModifiedValue(issueToUpdate.getCustomFieldValue(assetCf),
    assetCustomFieldValue), new DefaultIssueChangeHolder())
}

/**
 * Checks if Asset custom field value is empty
 */
def isCustomFieldEmpty(assetCustomFieldId) {
AssetCustomField assetCustomField = assetCustomFieldAndValue.getAssetCustomField();
if (assetCustomField.customFieldId == assetCustomFieldId) {
    if (assetCustomFieldAndValue.getAssetList() != null && assetCustomFieldAndValue.getAssetList().size() > 0) {
        return false;
    }
} return true;
}

try {
    if (isCustomFieldEmpty(assetCustomFieldId)) {
        def assetCfValue = getAssetFieldValue(AUTH_HASH, assetTypeName, fieldName, fieldValue)
        if (assetCfValue != null) {
            updateAssetCustomFieldOfTheIssue(assetCustomFieldId, issue.getKey(), assetCfValue)
        }
        return assetCfValue
    }
} catch (Exception e) {
    logger.error("Error while setting asset custom field value.", e);
}
return null;

Save post function and move it after "Creates the issue originally."

Publish post function and test it.

Test.1 - Asset field is empty
<table>
<thead>
<tr>
<th>Create issue</th>
<th>View Issue</th>
</tr>
</thead>
<tbody>
<tr>
<td>User is tyler and Computer field is blank.</td>
<td>Computer value is automatically set to &quot;Computer 01&quot; where it's JIRA User attribute value is tyler.</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Summary</td>
<td></td>
</tr>
<tr>
<td>Computer field will be left blank (user is tyler)</td>
<td></td>
</tr>
<tr>
<td>Computer (optional)</td>
<td></td>
</tr>
<tr>
<td>Start typing to search by name or #ID (e.g., #123)</td>
<td></td>
</tr>
<tr>
<td>Description (optional)</td>
<td></td>
</tr>
<tr>
<td>test</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>[Create]</td>
<td></td>
</tr>
<tr>
<td>[Cancel]</td>
<td></td>
</tr>
</tbody>
</table>

Test.2 - Asset field is NOT empty

<table>
<thead>
<tr>
<th>Create issue</th>
<th>View Issue</th>
</tr>
</thead>
<tbody>
<tr>
<td>User is tyler and Computer field is not blank.</td>
<td>Computer value won't change</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
If tests are ok, it is ready for production.

Modify search

Following part of the script is search parameters. In this example it filters assets by Asset Type and one attribute value.

```python
def queryToPostJson = "    {"searchType" : "basic" ,
"listType" : "detail" ,
"queryIndexSearchParams" : [
            {
                "field" : "fieldName" ,
                "keyword" : "fieldValue" ,
                "keywords" : [] ,
                "fieldType" : "STRING" ,
                "range" : false
            },
            {
                "field" : "form.name" ,
                "keyword" : null ,
                "keywords" : ["assetTypeName"] ,
                "fieldType" : "LIST" ,
                "range" : false
            }
        ] ,
    "pageNumber" : 1 ,
    "pageSize" : 10000 ,
    "sortDirection" : "asc" ,
    "sortField" : "asset.id"
}"
```

You can modify it according to you needs, to find more alternatives, go to Asset Navigator and inspect.
In network tab, query.json is the service and locate Request Payload which is the search parameters.