ScriptRunner for JIRA plugin

This Plugin gives some extremely nice features, like:

- Using prebuilt-in functions/tasks
- Running scripts in Transitions.
- Online (GUI) groovy scriprunner

> With JIRA 7 the "ComponentManager" is deprecated, use "ComponentAccessor" instead.
>
> With JIRA 8 the "ComponentManager" is removed, use "ComponentAccessor" instead.

> The plugin has been purchased by Adaptavist in 2015 and is from version 4 Paid Product.

- Change Task Type
- Condition for Link
- Setting Issue values after "Create Subtask"
- Getting a Custom Field Value (String)
- Setting a Custom Field Value (String)
- Assigning a user
- Remove and Set labels
- Transist Linked Issues
- Get CurrentUser
- Send Custom Mail
- Making Fields Required
- Scripted Fields
- Links

Change Task Type

This actually gets hold on a clone and changes the "Task Type" Custom Field to "Clone", whereas on the Master it is still "Master"

```
import com.atlassian.jira.ComponentAccessor
import com.atlassian.jira.issue.customfields.manager.OptionsManager
import com.atlassian.jira.issue.CustomFieldManager

componentManager (componentManager) ComponentAccessor.getComponentManager()

optionsManager = (optionsManager) ComponentAccessor.getOptionsManager()
customFieldManager customFieldManager = ComponentAccessor.getCustomFieldManager()
def cf = customFieldManager.getCustomFieldObjects(issue).find {it.name == 'Task Type'}
def fieldConfig = cf.getRelevantConfig(issue)
def optionClone = optionsManager.getOptions(fieldConfig).find {it.value == "Clone"}
issue.setCustomFieldValue(cf, optionClone)
```

Condition for Link

This script sends "passesCondition = false" if there is an "Awaiting" Link type on the Issue.

```
import com.atlassian.jira.component.ComponentAccessor
import com.atlassian.jira.issue.link.IssueLink
import com.atlassian.jira.issue.link.IssueLinkManager
import com.atlassian.jira.issue.Issue
import org.apache.log4j.Category

IssueLinkManager issueLinkManager = ComponentAccessor.getIssueLinkManager()

String IssueType      = issue.getIssueType().name
String ProjectName    = issue.getProjectObject().name
String IssueKey       = issue.key
passesCondition = true

System.out.println("script=CheckTOPSLinkStatus.groovy Issuekey: " + IssueKey + " IssueType: " + IssueType + "
ProjectName: " + ProjectName + " action=StartScript")

//Traverse all links
issueLinkManager.getOutwardLinks(issue.id)?.each {issueLink ->
   String LinkTypeName    = issueLink.issueLinkType.name
   String LinkStatus      = issueLink.getDestinationObject().getStatus().name
   String LinkIssueKey    = issueLink.getDestinationObject().key
   System.out.println("script=CheckTOPSLinkStatus.groovy Issuekey: " + IssueKey + " Issue Link key: " +
LinkIssueKey + " Issue Link Type: " + LinkTypeName + " Linked Issue Status: " + LinkStatus)
   if (LinkTypeName == "Awaiting")
   {
        passesCondition = false
        System.out.println("script=CheckTOPSLinkStatus.groovy Issuekey: " + IssueKey + " IssueType: " +
IssueType + " ProjectName: " + ProjectName + " action=Hide Transition")
   }
}
System.out.println("script=CheckTOPSLinkStatus.groovy Issuekey: " + IssueKey + " IssueType: " + IssueType + "
ProjectName: " + ProjectName + " action=EndScript")
```

Setting Issue values after "Create Subtask"

In This sample "transientVars" refers to the parent values

See issue properties/methods reference

```
issue.assignee == transientVars["issue"].assignee;
issue.setAssignee(null);
issue.setDescription("A new Description");
issue.setSummary("Unit Test - " + transientVars["issue"].key)
```

Getting a Custom Field Value (String)

```
import com.atlassian.jira.ComponentAccessor;
import com.atlassian.jira.issue.fields.CustomField
import com.atlassian.jira.issue.CustomFieldManager
customFieldManager customFieldManager = ComponentAccessor.getCustomFieldManager()
CustomField UseCustomer = customFieldManager.getCustomFieldObject(10600)

System.out.println("Custom field value: " + issue.getCustomFieldValue(UseCustomer))
```

Setting a Custom Field Value (String)

Taken the value fetched above:

```
issue.setCustomFieldValue(Customer,issue.getCustomFieldValue(UseCustomer))
```

This does not work for all field types (like labels).

Assigning a user

```
String userName="pho"
def userManager = ComponentAccessor.getUserManager()
def user = userManager.getUserObject(userName)
issue.setAssignee(user)

//Update the issue - may not be nessesary
ComponentManager.getInstance().getIssueManager().updateIssue(ComponentManager.getInstance().
jiraAuthenticationContext?.user, issue, EventDispatchOption.DO_NOT_DISPATCH, false)
ComponentManager.getInstance().getIndexManager().reIndex(issue)
```

A broader picture (ref: https://answers.atlassian.com/questions/98433/assigning-issues-via-groovy):

```
import com.atlassian.jira.ComponentAccessor
import com.atlassian.jira.security.Permissions
import com.atlassian.jira.event.type.EventDispatchOption
import com.atlassian.jira.util.ImportUtils

//Examples, 8 is my subtask issue type id. Will be different for others
updateAssignee("PRJ-1485", "user1")

//Method to do all the work
def updateAssignee(parentId, userName) {
def issue = ComponentAccessor.getIssueManager().getIssueObject(parentId)
def userManager = ComponentAccessor.getUserManager()
def user = userManager.getUserObject(userName)
issue.setAssignee(user)
//Update the issue
ComponentAccessor.getIssueManager().updateIssue(ComponentAccessor.jiraAuthenticationContext?.user, issue,
EventDispatchOption.DO_NOT_DISPATCH, false)
ComponentAccessor.getIndexManager().reIndex(issue)
}
```

Remove and Set labels

```
import com.atlassian.jira.ComponentAccessor
import com.atlassian.jira.issue.fields.CustomField
import com.atlassian.jira.issue.CustomFieldManager
import com.atlassian.jira.issue.label.LabelManager
import org.springframework.util.StringUtils
import java.util.Arrays
import java.util.HashSet
import java.util.Set

labelManager = ComponentAccessor.getLabelManager()
CustomFieldManager customFieldManager = ComponentAccessor.getCustomFieldManager()


CustomField UseCustomer = customFieldManager.getCustomFieldObject(10600)
CustomField Customer = customFieldManager.getCustomFieldObject(10900)

labelManager.removeLabelsForCustomField(10900)

Set labelSet = new HashSet();
labelSet.add(issue.getCustomFieldValue(UseCustomer))

def authContext = ComponentAccessor.getJiraAuthenticationContext()
def user = authContext.getUser()

// issue.getCustomFieldValue(UseCustomer)
// Does not work - the "labelSet" is not a valid set - https://answers.atlassian.com/questions/332748/set-
labels-in-jira
labelManager.setLabels(user, issue.id,labelSet, false,false)
```

Transist Linked Issues

This is very nice, for "auto" progressing linked issues. (Idea and source from this)

```
import com.atlassian.jira.component.ComponentAccessor
import com.atlassian.jira.issue.link.IssueLink

def issueLinkManager = ComponentAccessor.getIssueLinkManager()

import org.apache.log4j.Category
import com.atlassian.jira.issue.comments.CommentManager
import com.opensymphony.workflow.WorkflowContext
import com.atlassian.jira.workflow.WorkflowTransitionUtil
import com.atlassian.jira.workflow.WorkflowTransitionUtilImpl
import com.atlassian.jira.util.JiraUtils

def Category log = Category.getInstance("com.onresolve.jira.groovy.PostFunction")
log.setLevel(org.apache.log4j.Level.DEBUG)
log.debug "debug statements"

String currentUser = ((WorkflowContext) transientVars.get("context")).getCaller()
WorkflowTransitionUtil workflowTransitionUtil = ( WorkflowTransitionUtil ) JiraUtils.loadComponent(
WorkflowTransitionUtilImpl.class )


issueLinkManager.getOutwardLinks(issue.id)?.each {issueLink ->
        if (issueLink.issueLinkType.name == "parent-child") {
        // Transist Issue
                workflowTransitionUtil.setIssue(issueLink.getDestinationObject());
                workflowTransitionUtil.setUsername(currentUser);
                workflowTransitionUtil.setAction (741)    // 741 = Waiting
        // validate and transition issue
                workflowTransitionUtil.validate()
                workflowTransitionUtil.progress()
        // Add a comment so people have a clue why the child has been closed
                CommentManager commentManager = (CommentManager) ComponentAccessor.getCommentManager()
                String comment = "Status changed to *WAit for solution* as a result of the *Resolve* action
being applied to the TD Issue."
                commentManager.create(issueLink.getDestinationObject(), currentUser, comment, true)
       }
}
```



The

```
workflowTransitionUtil.setIssue(issueLink.getDestinationObject())
```

should be

```
workflowTransitionUtil.setIssue(issueLink.getSourceObject())
```

Going "the other way". There is also an getInwardLinks(issue.id) collection.

Get CurrentUser

```
import com.atlassian.jira.issue.Issue
import com.atlassian.jira.ComponentManager
User = ComponentAccessor.getJiraAuthenticationContext().getUser().getName()
```

Send Custom Mail

Some standard JIRA Field can be accessed very direct, other must be throught the

```
issue.getCustomFieldValue(componentManager.getCustomFieldManager().getCustomFieldObjectByName("Advisory
Subject"))
```

function.

Subject can be text or like:

```
($issue) <% out << issue.getCustomFieldValue(com.atlassian.jira.component.ComponentAccessor.
getCustomFieldManager().getCustomFieldObjectByName("Advisory Subject")) %>
```

Body can be like:

```
<% out << issue.getCustomFieldValue(com.atlassian.jira.component.ComponentAccessor.getCustomFieldManager().
getCustomFieldObjectByName("Mail Body")) %>
<br><br>
Best Regards,
<br><br>
GService Desk
<br>
<a href="http://sd.mydomain.dk">Service Desk</a>
<br>
You cant reply to this email.
```

Making Fields Required

This script is a sample from a Validation Function

```
// This script makes a requirement: if the Custom Field "OprStatusShow" is set to public, Incident Start and
Resolved must be set.

import com.opensymphony.workflow.InvalidInputException
import com.atlassian.jira.project.ProjectManager
import com.atlassian.jira.issue.CustomFieldManager
import com.atlassian.jira.issue.fields.CustomField

//Project "Support" Id = 10130
if (issue.getProjectObject().getId() == 10130) {

CustomFieldManager customFieldManager = ComponentAccessor.getCustomFieldManager()
CustomField IncidentResolvedField = customFieldManager.getCustomFieldObject("customfield_12224")
CustomField IncidentStartField = customFieldManager.getCustomFieldObject("customfield_10091")
CustomField OprStatusShowField = customFieldManager.getCustomFieldObject("customfield_11820")


String IncidentResolved = (String)issue.getCustomFieldValue(IncidentResolvedField)
String IncidentStart = (String)issue.getCustomFieldValue(IncidentStartField)
String OprStatusShow = (String)issue.getCustomFieldValue(OprStatusShowField)

if (OprStatusShow == "Yes") {

        if (IncidentStart == null || IncidentResolved == null) {
                InvalidInputException e= new InvalidInputException();
                e.addError("Incident Start And Incident Resolved must not be empty")
                 throw e
                }
        }
}
```

Scripted Fields

A "neat" but raw idea for a scripted field: https://answers.atlassian.com/questions/191893

Links

https://jamieechlin.atlassian.net/wiki/display/GRV/Post+Functions

http://quisapps.com/confluence/display/JSS/Scripting+Samples