Using Generic User Properties to Implement the Four Eyes Principle in IBM Business Automation Manager Open Editions (IBM BAMOE)

Author: Courtney Silva (csilva@salientprocess.com)

The four eyes principle states that two people must validate or approve an activity, event, result, or decision. These two people act as controllers and usually take shared responsibility for validation or approval. Therefore, the four eyes principle is a method of quality control with the aim of confirming quality, reducing errors, or avoiding misuse.

From a business process perspective, this means that whoever approved/completed the first task, cannot approve/complete the second task in an approval chain.

In this guide we will cover how to implement this practice with IBM Business Automation Manager Open Editions (BAMOE), formerly known as Red Hat Process Automation Manager (RHPAM). Implementing the four eyes principle with IBM BAMOE is made easy due to generic user properties that you can leverage in your process.

> TIP: To see a list of generic user properties, refer to the generic user properties section here.

Four Eyes Principle Practical Example

To implement a business process that reflects an approval process based on the good practices mentioned, we will explore a business process that contains two Human Tasks. Remember that our goal is to make sure that even though both users belong to the same approver's group, each task must be completed by a different user.

If put in simple terms, the user who completed the first task will be excluded from the group of potential owners of the second approval task.

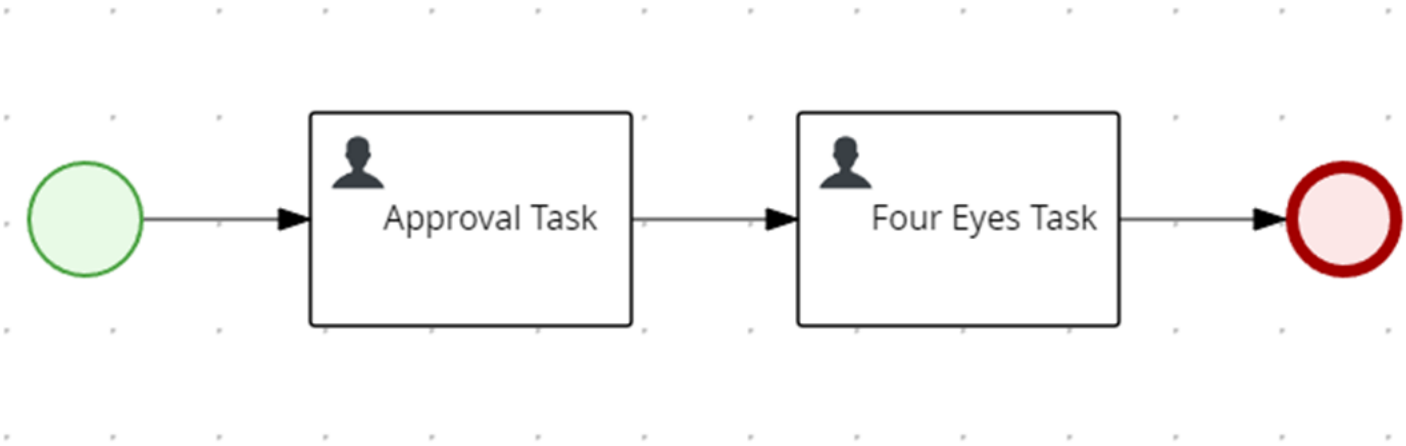

Figure 1. Four eyes principle process implementation

First scenario: Single Approver

Initially, we'll start by handling a single approver user. These are the steps we'll go through:

1. Configure a process variable named "approver" to hold the user who completed the "Approval Task";
2. Once the "Approval Task" is completed, fill in "approver" variable value using the out-of-the-box variable "`ActorId`";
3. Exclude the "approver" user from the potential owners of the task "Four Eyes Task", using another BAMOE variable, "**`ExcludedOwnerId`**";

"**`ActorId`**" and "**`ExcludedOwnerId`**" are generic user properties, where "`ActorId`" is the performer of the task to whom the task is assigned, and "`ExcludedOwnerId`" is anyone who has been excluded to perform the task and become an actual or potential owner.

Now, let's drill down on the details of this implementation in BAMOE:

1. In this case, both tasks are assigned to the "Approvers" group, which includes users "foureyesuser1", "foureyesuser2", and "foureyesuser3".

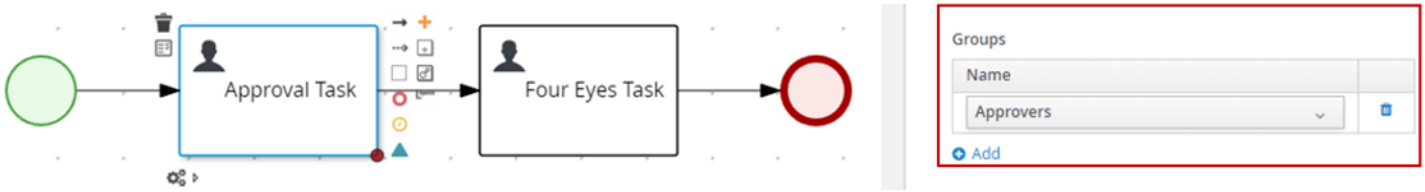

Figure 2. The "Approvers" group assigned to the "Approval Task" and the "Four Eyes Task"

2. In the process configurations, we will use the process variable named "approver" to store the user who is acting on the first task - and should not be working on the "Four Eyes Task".

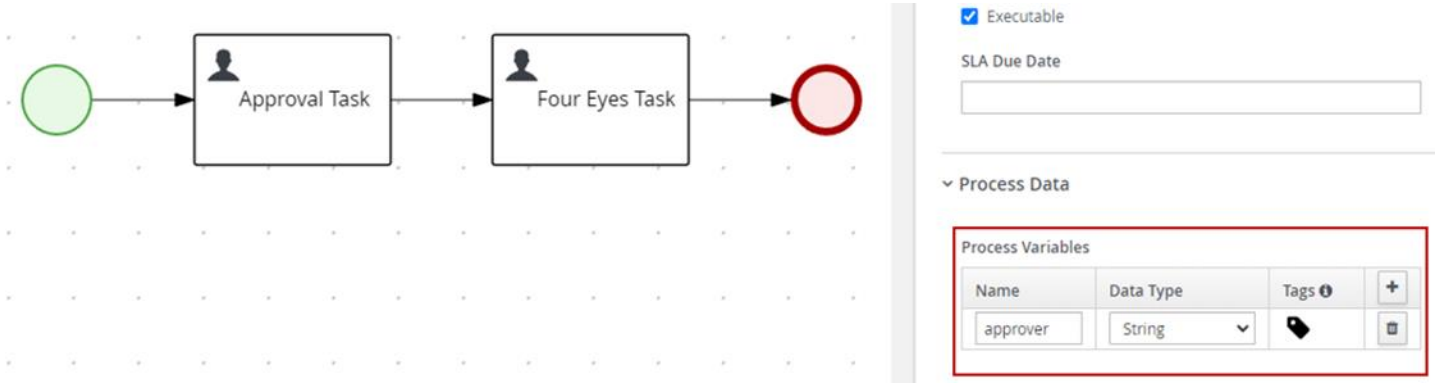

Figure 3. The "approver" process variable

3. In the "Approval Task", the data output of the "approver" process variable will hold the value of "ActorId".

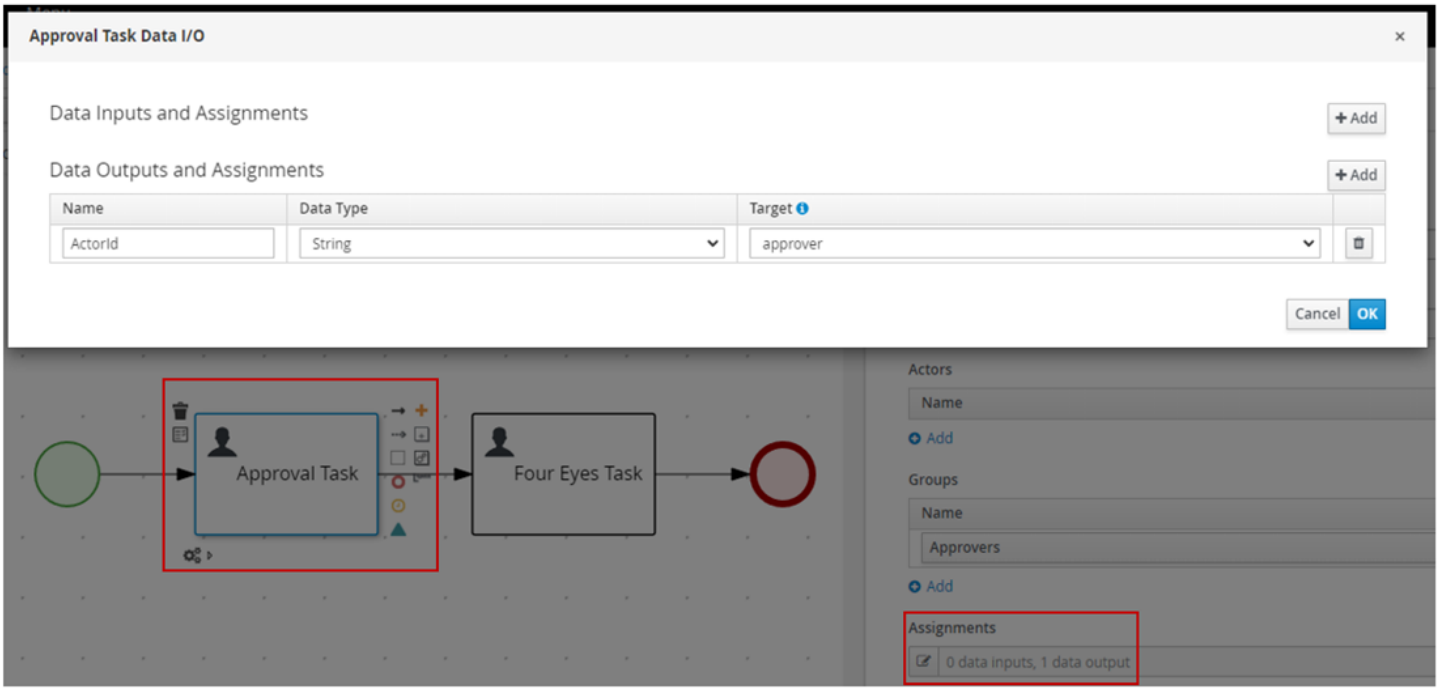

Figure 4. The Data Outputs and Assignments for the “Approval Task”

3. The “Four Eyes Task” data input of the “approver” process variable will set the value for “ExcludedOwnerId”.

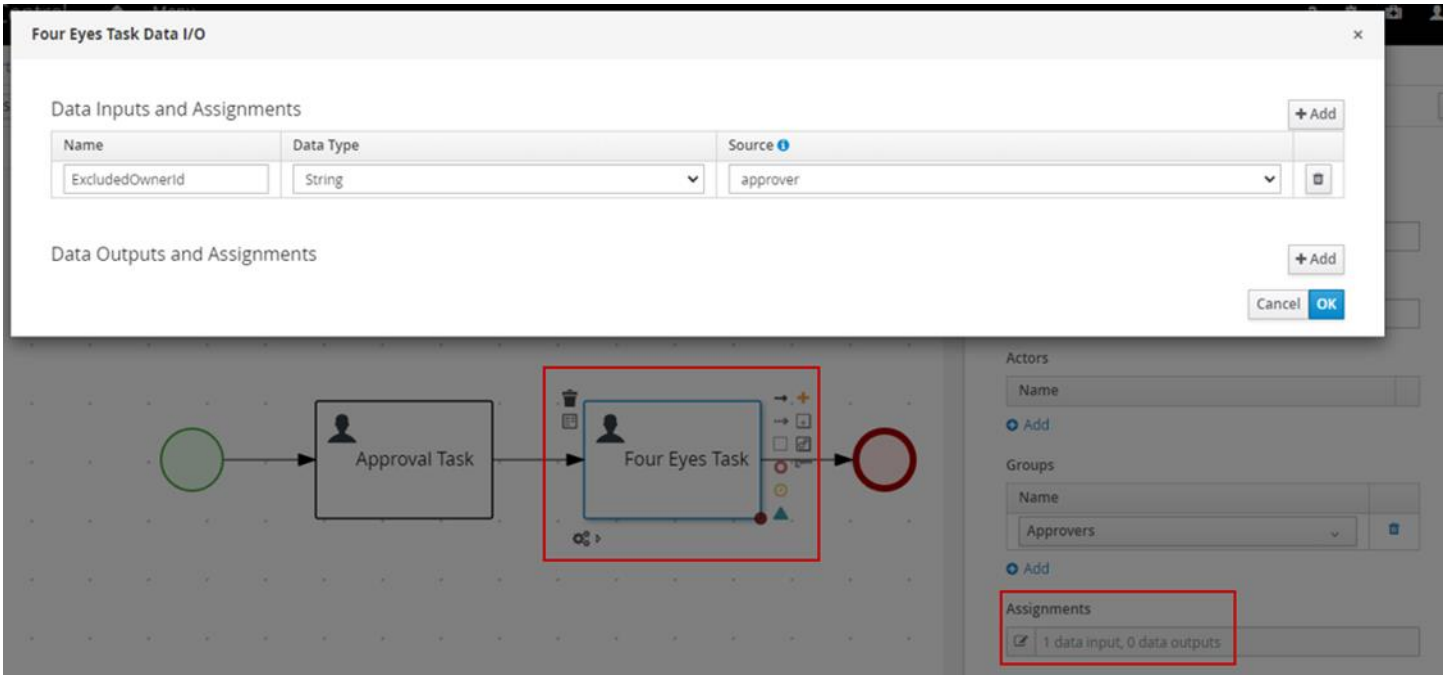

Figure 5. The Data Inputs and Assignments for the “Four Eyes Task”

When testing this process, you'll notice that at runtime, the owner who completes the “Approval Task” will not see the “Four Eyes Task” in their task inbox.

Second Scenario: Multiple Approvers

In this second example, you'll notice that removing multiple owners from a task is as simple as removing a single owner. The only difference is passing a comma separated String to "`ExcludedOwnerId`". For learning purposes, we will simplify the process by using a Script Task to configure the list of users we want to remove from the list of people who can approve the following task. This sample process looks like:

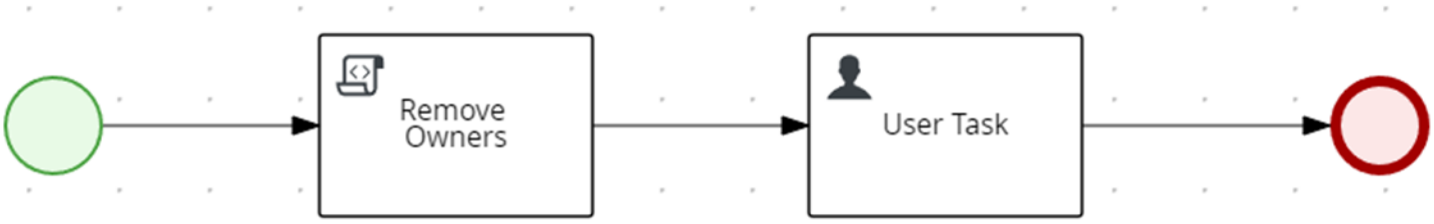

Figure 6. Multiple excluded owners process implementation

1. Assign the "User Task" to the "Approvers" group. In our example, this group includes users "foureyesuser1", "foureyesuser2", and "foureyesuser3".

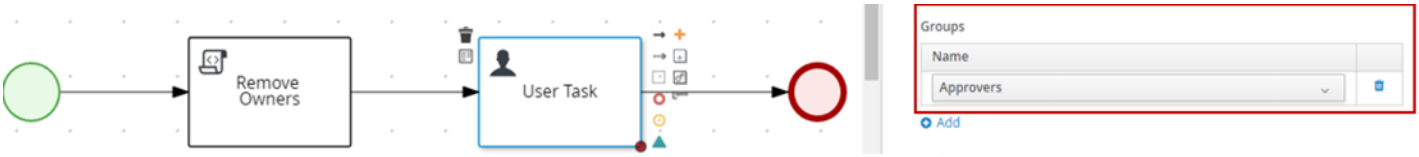

Figure 7. The "Approvers" group assigned to the "User Task"

2. Use a process variable to store the excluded user ids. In our example, we'll name the process variable as "`excludedOwners`".

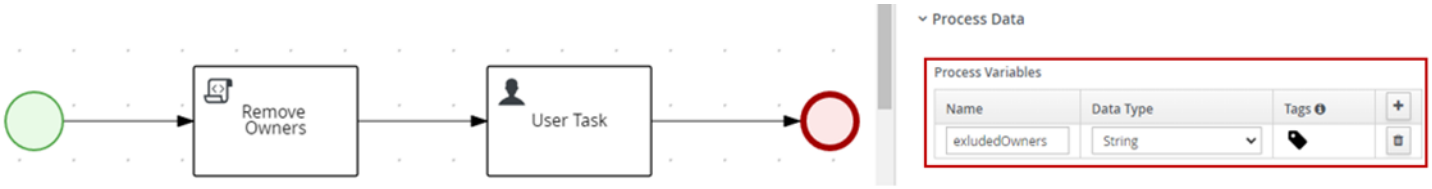

Figure 8. The "excludedOwners" process variable

3. In our example, we'll manually set a list of users. You can have as many approval tasks as needed for your use case, and store each "ActorId" on a list. Notice below that the value of "excludedOwners" is set in the "Remove Owners" script task.

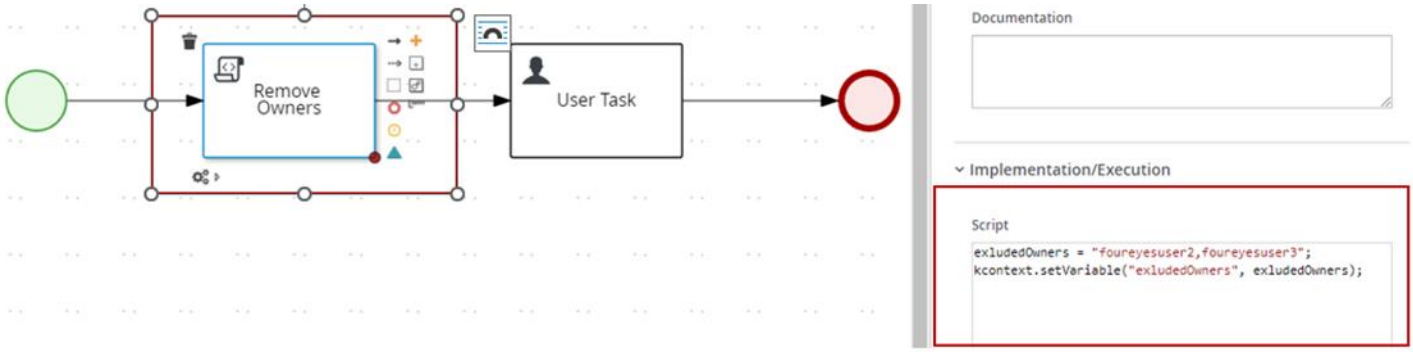

Figure 9. “Remove Owners" script task that adds users “foureyesuser2”, and “foureyesuser3” to process variable “excludedOwners”

4. The “`excludedOwners`” process variable is then passed as an input to the “User Task”. It will set the value for “ExcludedOwnerId”.

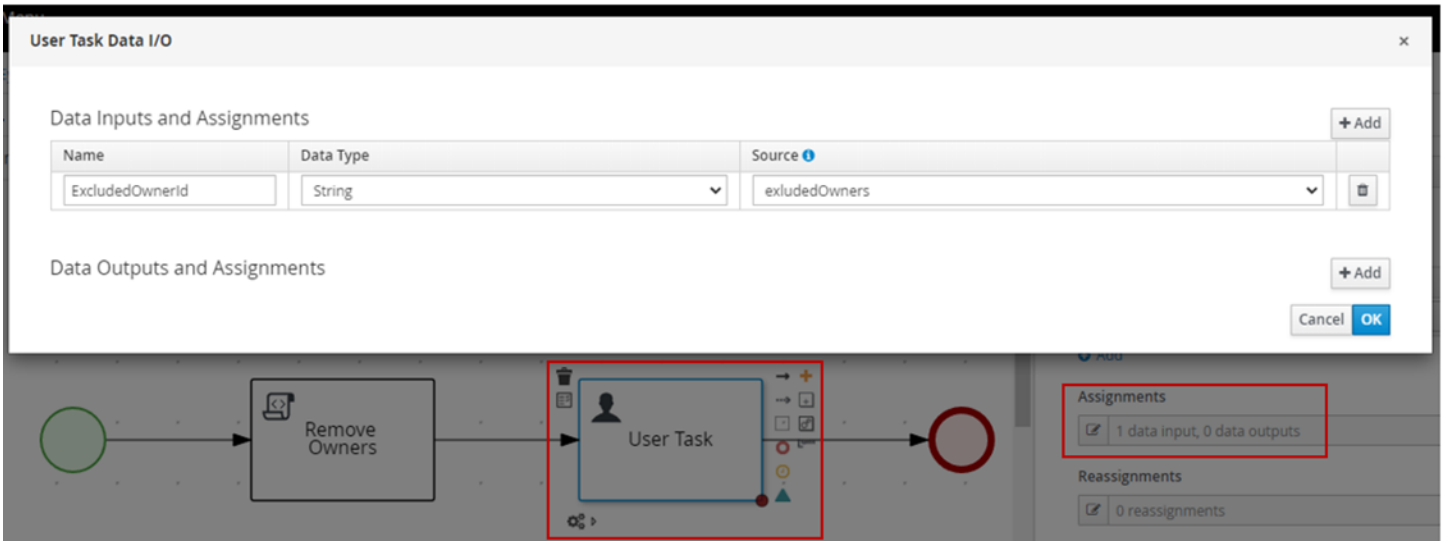

Figure 10. The Data Inputs and Assignments for the “User Task”

When testing this business process, you'll notice that at runtime, the engine will be able to manage the task owners properly where “foureyesuser2”, and “foureyesuser3” will not be able to claim the “User Task”.

It is important to highlight the different process behavior when you pass multiple excluded owners to “`ExcludedOwnerId`” as opposed to a single excluded owner: since “foureyesuser1” is not an excluded owner, the user can claim and complete the task as expected. Although, when working with multiple excluded owners, be aware that these users will still be able to see the task in their task inbox, however, **they are unable to claim it**. Figures 11 and 12 show the behavior in Business Central, and the same can be expected if you interact with the process engine (KIE Server) via REST API.

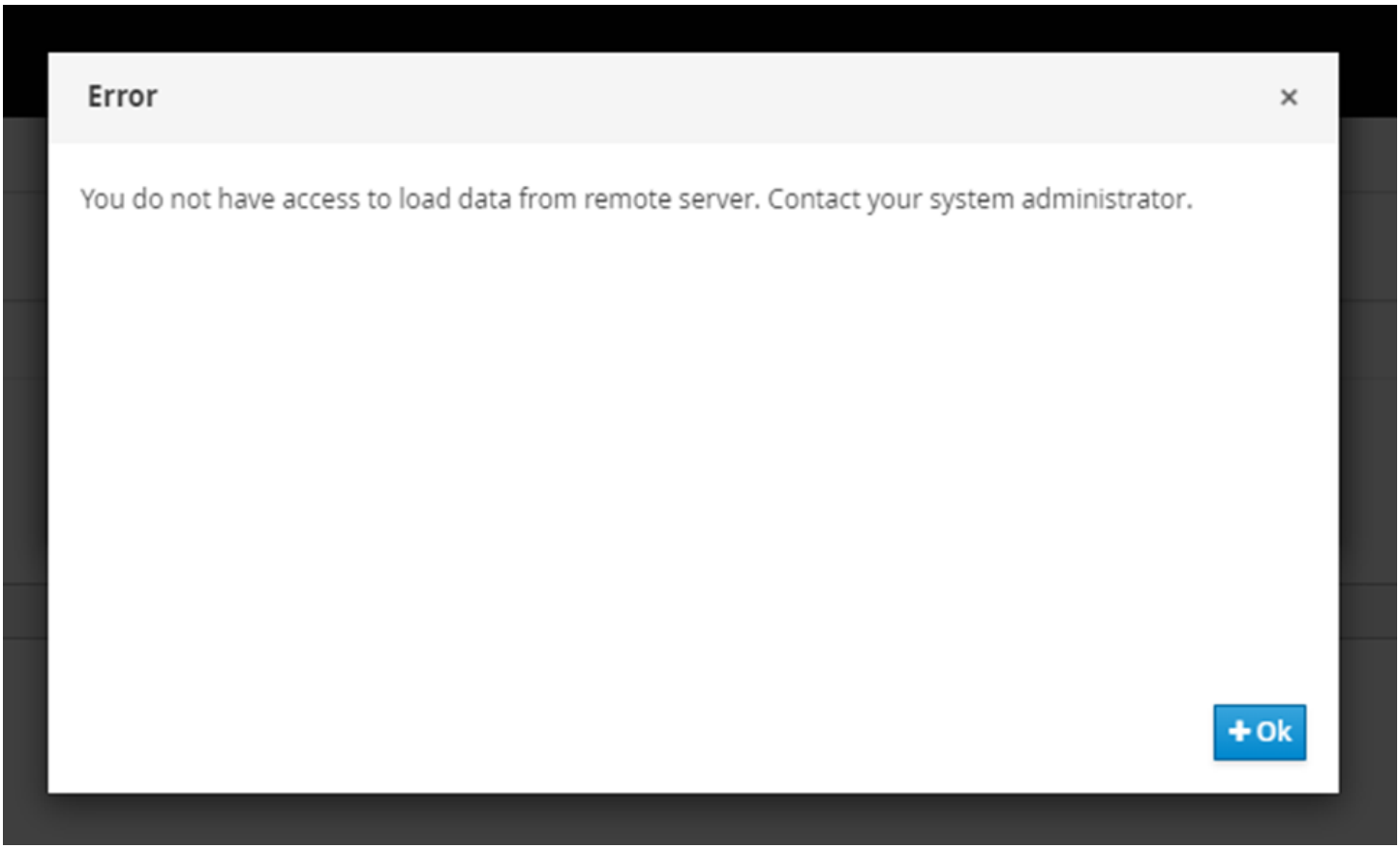

Figure 11. The error “foureyesuser2”, and “foureyesuser3” see when trying to claim the “User Task”

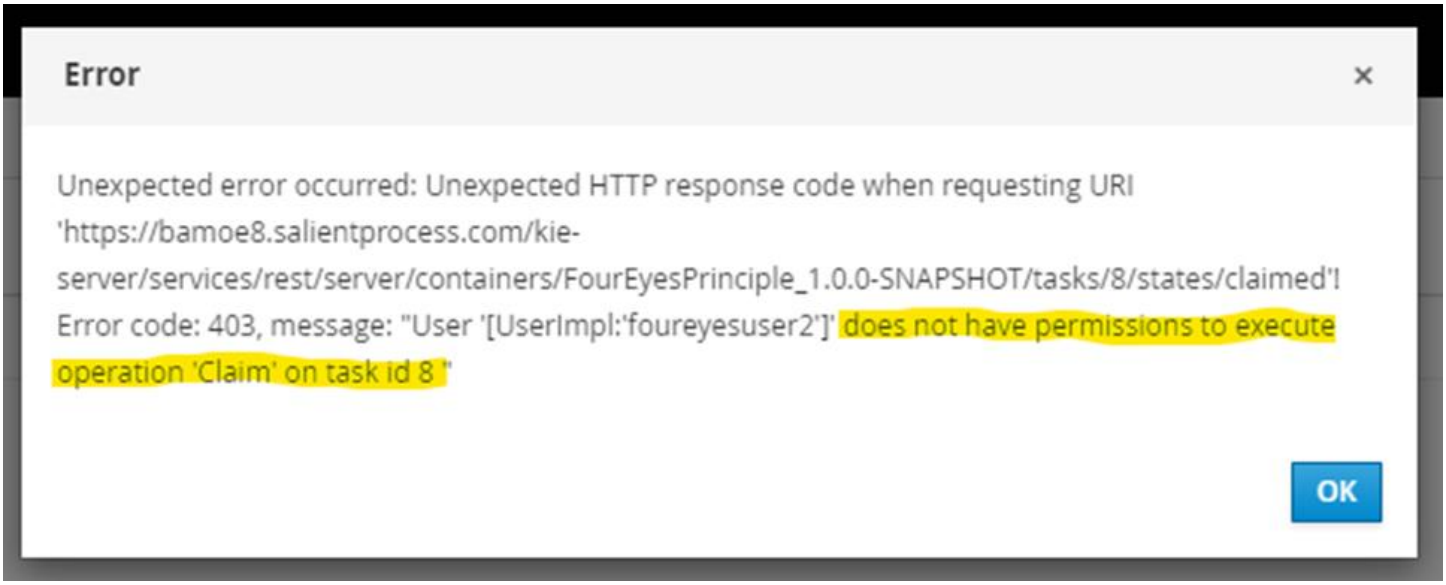

Figure 12. Exception is thrown when the user clicks on the “+Ok” button represented on figure 11

About the Generic User Properties

- **ActorId**: The performer of the task to whom the task is assigned.
- **GroupId**: The group to which the task performer belongs.
- **BusinessAdministratorId**: The default business administrator responsible for the progress and the outcome of a task at the task definition level.
- **BusinessAdministratorGroupId**: The group to which the administrator belongs.

- **ExcludedOwnerId**: Anybody who has been excluded from performing the task and become an actual or potential owner.
- **RecipientId**: A person who is the recipient of notifications related to the task. A notification may have more than one recipient.

If you need to override the default values of these generic user properties, you must define a data input with the name of the property, and then set the desired value in the assignment section.

Conclusion

The Four Eyes Principle, which calls for shared responsibility in task validation or approval, can be effectively implemented with IBM Business Automation Manager Open Editions. By leveraging the generic user properties such as "ActorId" and "ExcludedOwnerId", you can adopt this principle on your business processes to enhance quality assurance, reduce errors, and prevent misuse.

Remember that you can rely on the engine capabilities for scenarios with one or multiple approvers. BAMOE can serve as a vital tool for automated workflows that seek to better quality control and validation procedures, with strengthened trust and accountability.

If you have further questions or would like more information, please contact Salient's Automation Advisors. They will provide you with detailed information, expert guidance, and support in this area.