DATAGROUP

1. APROVAL AH EXAMPLE:

```
do

#Approval Model
LOCAL::model = '{
  "action":"approve",
  "task":"The task needs to be approved (Email Title)",
  "template":"Please Accept or Reject the request (Email Body)",
  "controls":[
      {
        "approve":{"/approve":"true"},
        "reject":{"/approve":"false"}
      },
      {
        "text": ["/Name", "/Surname"]
      }
    ]
  }'


#Execution AH
  exec: LOCAL::RESULT,
  stdout: LOCAL::OUTPUT,
  stderr: LOCAL::ERROR,
  exit: LOCAL::SYSTEMRC = action(
      capability: "AskApprove",
      user_email: emailAddress,
      issue_id: ISSUE::ogit/_id,
      question_model: LOCAL::model
      )

  log.info(LOCAL::RESULT)
  log.info(LOCAL::OUTPUT)
  log.error(LOCAL::ERROR)
  log.info(LOCAL::SYSTEMRC)


#Result interpretation
  if LOCAL::SYSTEMRC == 0 then
    log("SUCCESS: ${LOCAL::OUTPUT}")
  else
    log.error("ERROR ${LOCAL::ERROR}")
  end

#Set the task into Waiting state for external event
  eject_issue(message: "Wait for approve", issue_id: ISSUE::ogit/_id)
```

In the AH:

- The capability of the AH must be set to **AskApprove**.
- The **user_email** represents the recipient´s email address where the approval request will be sent.
- The issue_id stores the ogit/id of the current issue.
- The questions structure is defined in **LOCAL::model** (details below)
- Before sending the approval email, the KI must call **eject_issue**. This action transitions the Task into a WAITING state. The Approval user interface will then update the KI to PROCESSING. If the Task is not set to WAITING, it will not appear in the user interface.

This AH requires a model definition, which we will now explain in more details.

2. MODEL

The model serves as the interface presented to the approver, encompassing key components such as the approval email title, approval email body message, and interactive controls. It allows for customization with diverse layouts for controls, ensuring mandatory inclusion of "Approve" and "Reject" buttons. Additional configurable options include text boxes for input and drop-down menus for selecting specific choices.

```
LOCAL::model = '{
  "action":"approve",
  "task":"${approvalTitle}",
  "template":"${approvalBody}",
  "controls":[
      {
        "approve":{"/approve":"true"},
        "reject":{"/approve":"false"}
      },
      {
        "text": ["/Name", "/Surname"]
      }
    ]
  }'
```

- "action": "<ACTION>"

  This specifies the user-friendly name of the action the user will perform. For the approval AH, this value must always be "approve". No other value is allowed.

- "task": "<TASK>"

This defines the user-friendly name of the task associated with the action, such as "Add User to Security Group". This parameter will be used as the task title in both email sent to the approver and in the user inteface.

- "template": "<TEMPLATE>"

  This provides a user-friendly description of the task. It will serve as the message body content in both email sent to the approver and in the user interface.

- "controls": [<CONTROL>, <CONTROL>, …]

  The array of questions / inputs the user needs to provide. The controls array can contain one or more <CONTROL> objects. These can be:

  - The "Approve" or "Reject" objects are mandatory controls and must be selected first (appearing in botht the email sent to the approver and in the user interface).
  - Once selected, other control objects like **text boxes** or **dropdown menus** can become available(only visible in the use interface).
  - Any data entered in these fields will be saved directly to the automation issue for further processes.

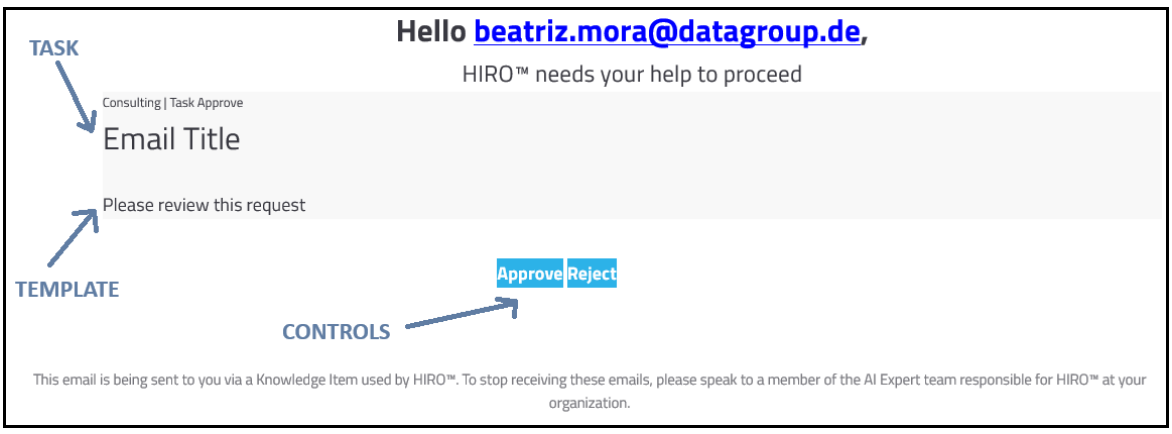
The action parameter is directly linked to the Approval AH and no other value is allowed.The remaining three parameters enable customization of the approval request.

EMAIL FORMAT

This is the email the approver will receive to review the request and provide any necessary information, if required.

The email format for the approval request will include

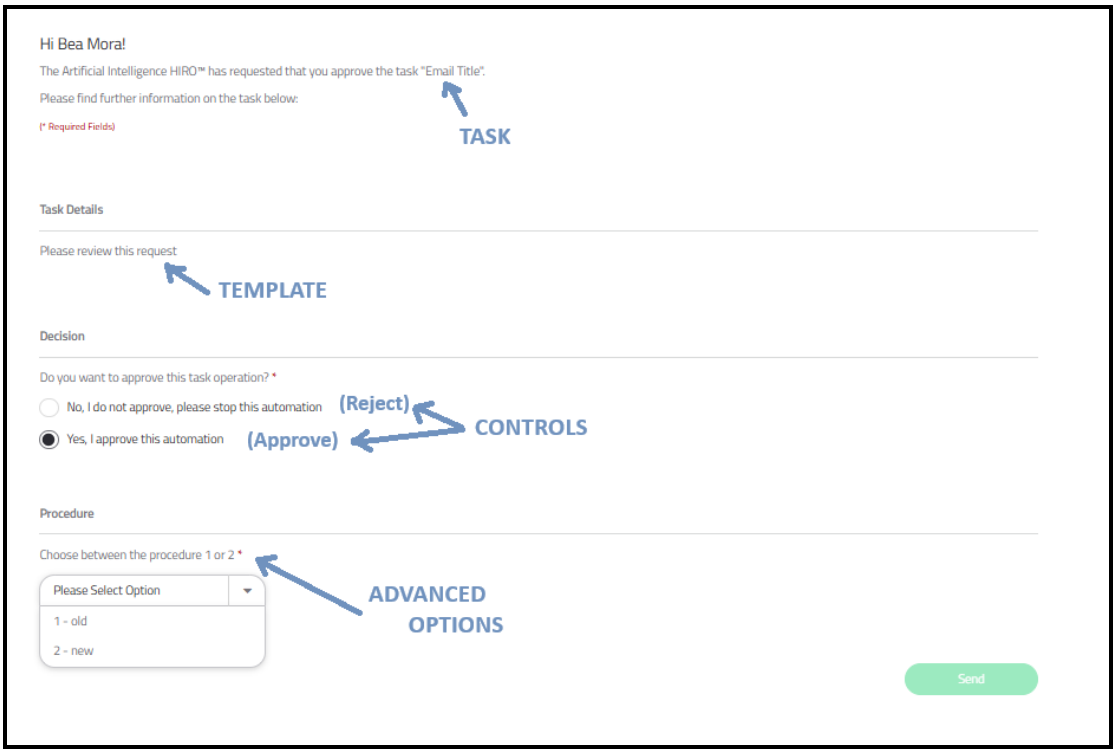
- Title and content: Reflects the ¨task¨ and ¨template¨ details provided within the model.
- Mandatory Controls: ¨Approve¨ and ¨Reject¨ buttons as mandatory steps to secure an approval.



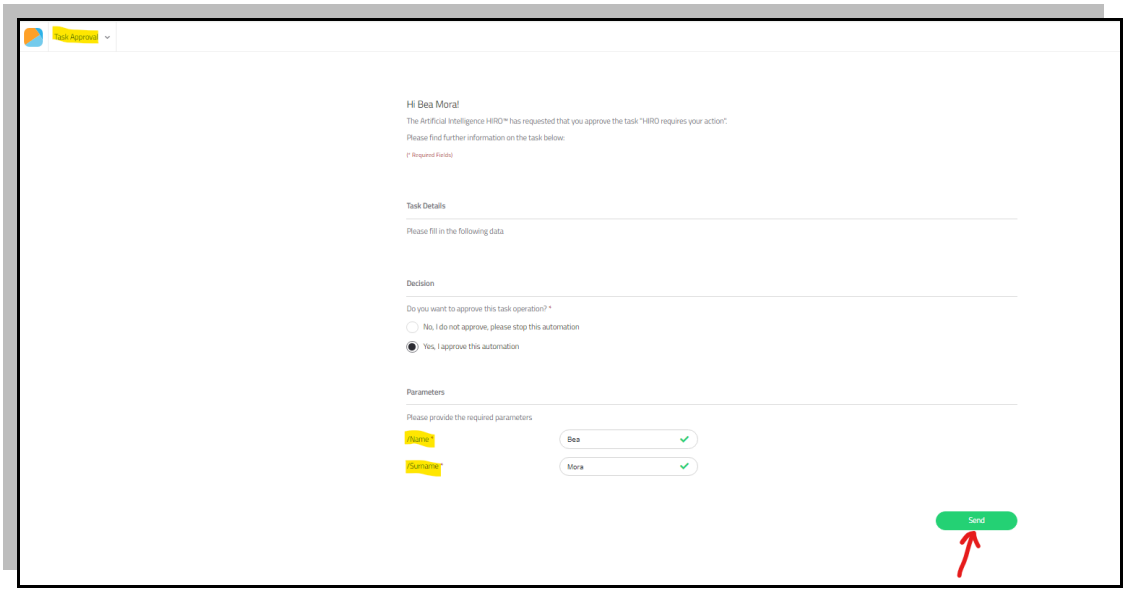
TASK APPROVAL(User interface)

Upon selecting ¨approve¨ or ¨reject¨ in the email, you will be seamlessly directed to the "Task Approval" application within the HIRO desktop. Here, you can reconfirm or reject the request and input any additional information, such as Name and Surname, as specified in the first model. The injection of this data into HIRO is exclusively managed within the HIRO environment. It is important to note that this data injection cannot be performed via email.

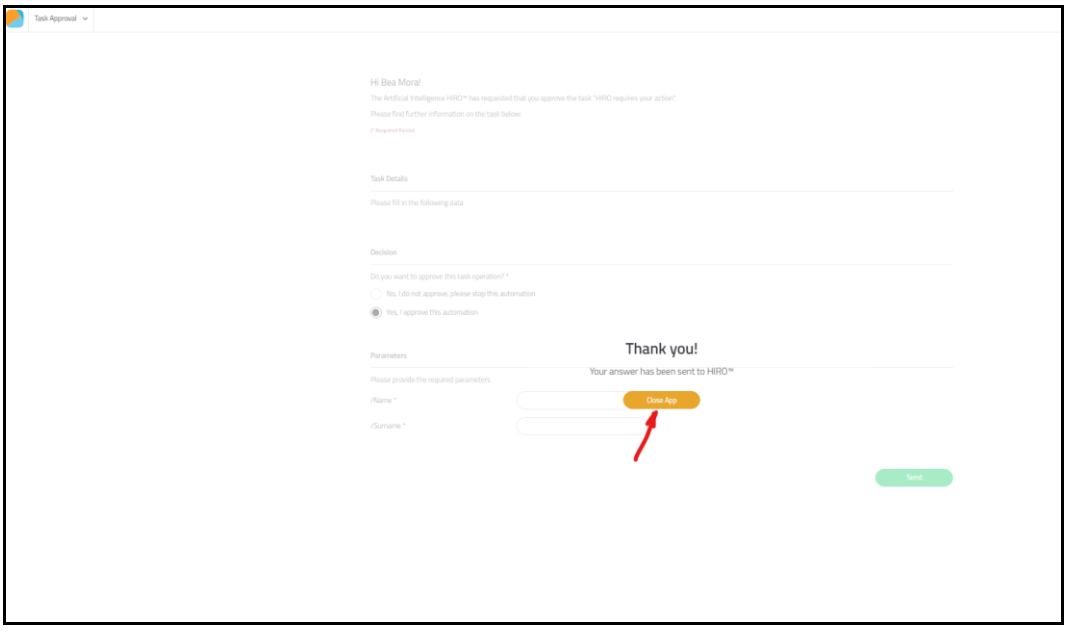
In the user interface, you can view all the attributes from the model, including ¨task¨, ¨template¨, mandatory ¨controls¨ and the advanced options that appear after approving or rejecting the request (these advanced options are only available in the user interface).



This is the "Task Approval" app within HIRO desktop.



Upon clicking ¨Send¨ to communicate your decision, the application will automatically close. As the approver, no further action is required at this point.



MODEL VARIATIONS

1. MODEL: STORE INFORMATION IN SEVERAL VARIABLES FOR ONE OF THE CONTROLS

```
LOCAL::model = '{
  "action":"approve",
  "task":"${approvalTitle}",
  "template":"${approvalBody}",
  "controls":[
      {
        "approve":{"/approve":"true"},
        "reject":{"/approve":"false" , "/email":"${approverEmail}" }
      }
    ]
  }'
```

In this model, while storing the value for the "approve/reject" control, we can also store other useful variables that might assist in further task resolution.

- `"approve"`: Holds the DataObject to be saved on approval.
- `"reject"`: Holds the DataObject to be saved on rejection.

The DataObject's key is the attribute name ("/approve"), and the value is the data to be saved in the graph ("true" or "false"). The DataObject can be empty, or contain one or more attribute mappings.

2. MODEL: ACTIVATE A TEXT BOX FOR ONE OF THE CONTROLS (APPROVE OR REJECT).

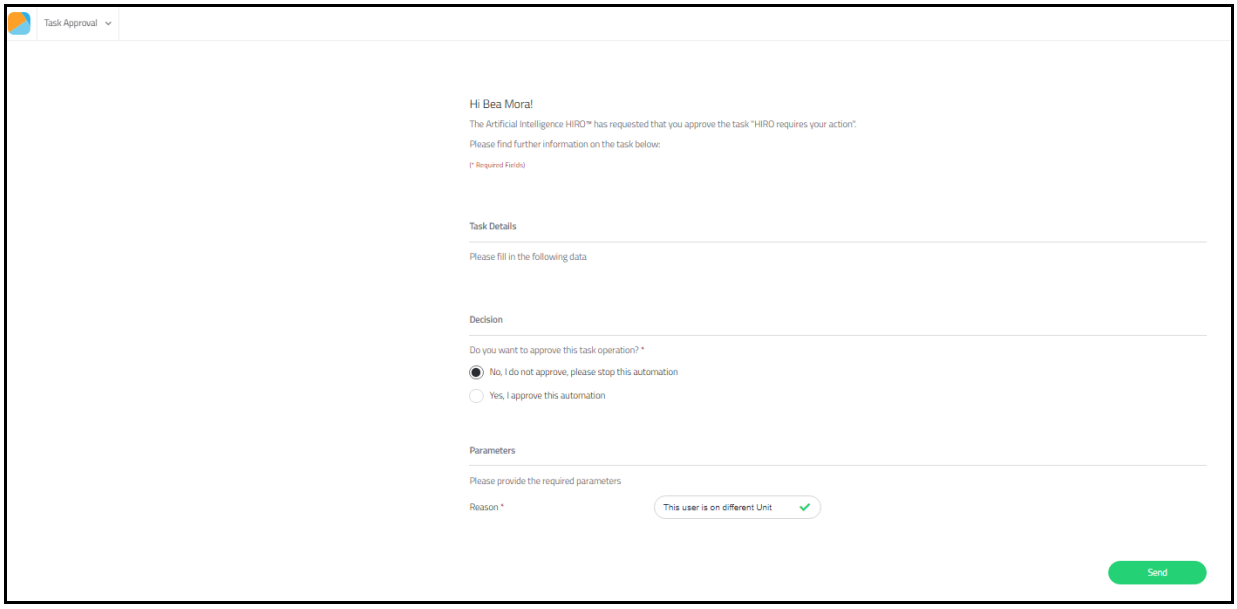
```
LOCAL::model = '{
  "action":"approve",
  "task":"${approvalTitle}",
  "template":"${approvalBody}",
  "controls":[
      {
        "approve":{"/approve":"true"},
        "reject":{"/approve":"false"}
      },
      {
        "text":{"Reason" : "/comments"},
        "for" : "reject"
      }
    ]
  }'
```

In this model, the text box will only appear if the approval is rejected. This feature enables users to provide feedback or reasons for rejection, which will be stored under the "comment" variable within HIRO. Users will only see the text field labeled "Reason" to fill in their comments.

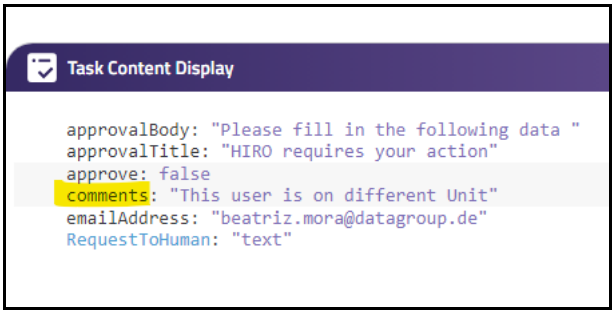
The Control Object Text can be used for both Approval and Rejection flows. By default, it is for Approval, but specifying the object key `"for"` allows its use in the Rejection flow.

- `"text"`: Holds the user´s input as a value under the attribute name (“/comments“). It could also hold an array of strings attributes names (Ex: “/ShortComments“ ; “/ActionNeededBy“).

IT’s that simple.



The "comment" variable is now visible on the Task with the value "This user is on different Unit" . This allows users to easily access and review any comments provided during the approval process.

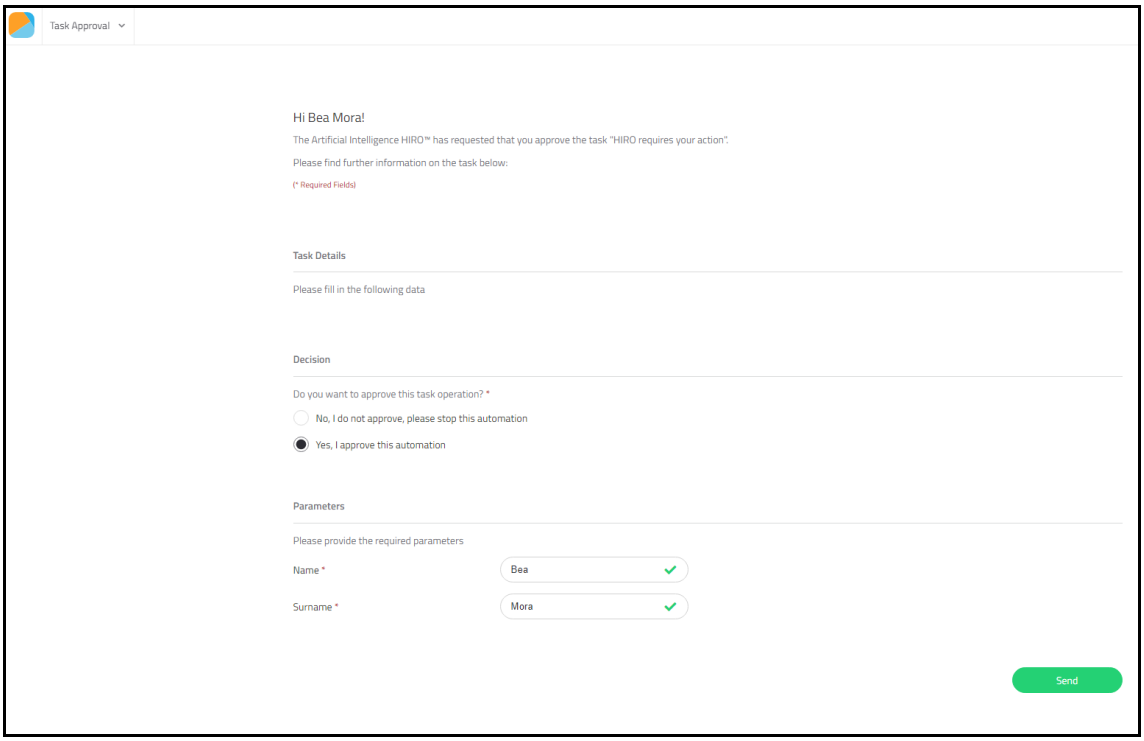


In this case, we are using the label "reason" for the text box. This label will appear on the HIRO user interface as shown in the screenshot above.

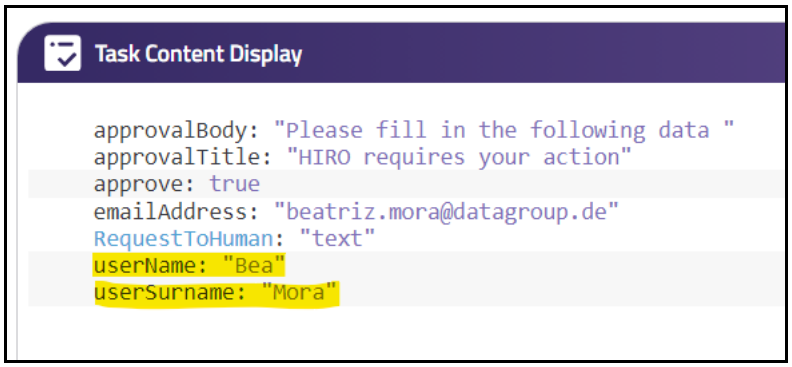
3. MODEL: ACTIVATE SEVERAL TEXT BOXES

```
LOCAL::model = '{
  "action":"approve",
  "task":"${approvalTitle}",
  "template":"${approvalBody}",
  "controls":[
      {
        "approve":{"/approve":"true"},
        "reject":{"/approve":"false"}
      },
      {
        "text": {"Name" : "/userName" , "Surname" : "/userSurname" },
        "for" : "approve"
      }
    ]
  }'
```

If you need to collect multiple pieces of information from the approver, you can utilize the text parameter. By structuring the text fields and variable names in JSON format, you can efficiently gather and store data within HIRO.



Now Name and Surname provided by the approver will be stored under variables "userName" and "userSurname" within HIRO.



4. MODEL: ACTIVATE A TEXT BOX USING THE VARIABLE NAME FOR BOTH THE TEXT FIELD AND VARIABLE ASSIGMENT IN HIRO

This way simplifies the process of inputting data directly into the variables used in HIRO.

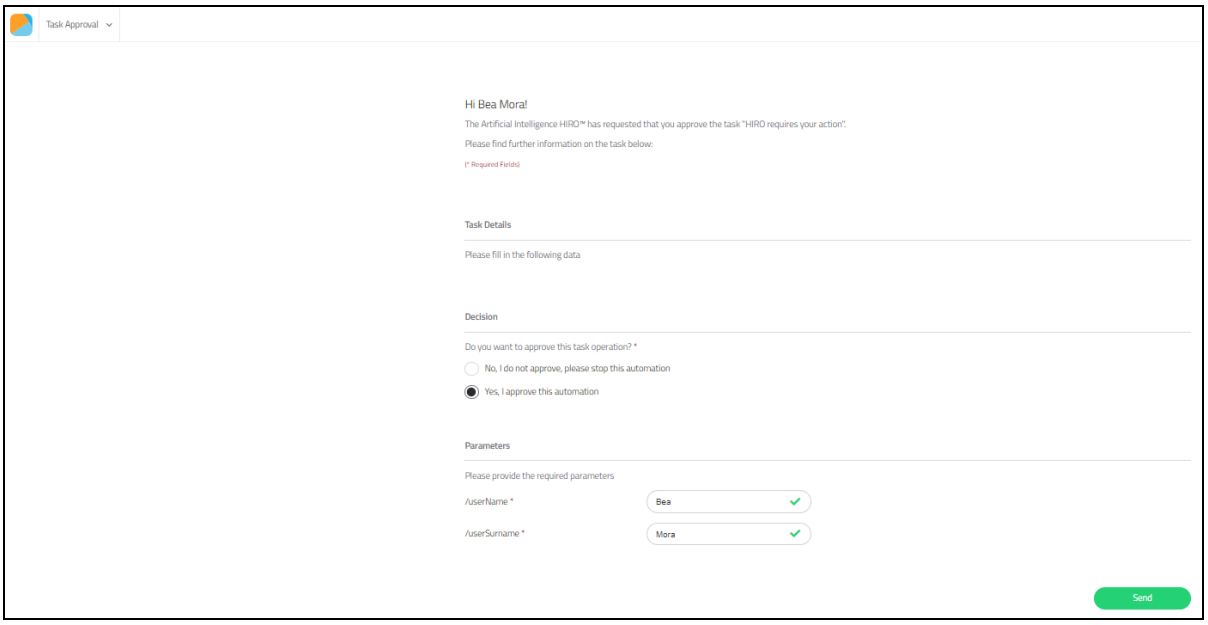
```
LOCAL::model = '{
  "action":"approve",
  "task":"${approvalTitle}",
  "template":"${approvalBody}",
  "controls":[
      {
        "approve":{"/approve":"true"},
        "reject":{"/approve":"false"}
      },
      {
        "text": ["/userName" , "/userSurname" ]
      }
    ]
  }'
```

You can also configure this text option to any of the mandatory controls (approve or reject).

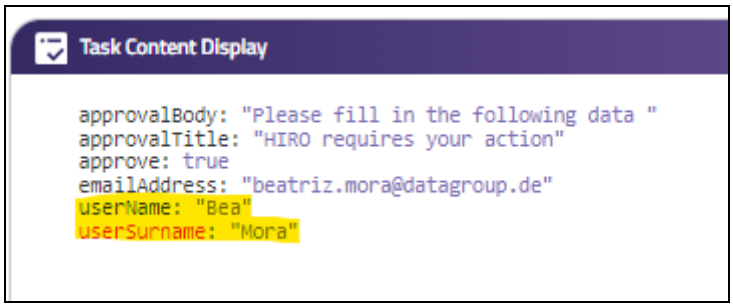
```
      {
        "text": ["/userName" , "/userSurname" ],
         "for" : "approve"
      }
```


IT's that simple.

In the Task Approval app, you can now immediately view the names of the variables that will store the data inputted by the approver.



This information is also displayed on the Task Content Display after KI execution



INPUT HEADER AND DESCRIPTION FOR TEXT CONTROL

By default, a header and description are provided for the Text control, but you can customize them using the following syntax:

```
{
  "text": ["/userName" , "/userSurname" ],
  "for" : "approve",
  "header": "Personal Info",
  "desc": "Please introduce the following details"
}
```


IT's that simple.

This results in the following layout:



- `"header"`: Optional Header text for this section.
- `"desc"`: Optional Description text for this section.

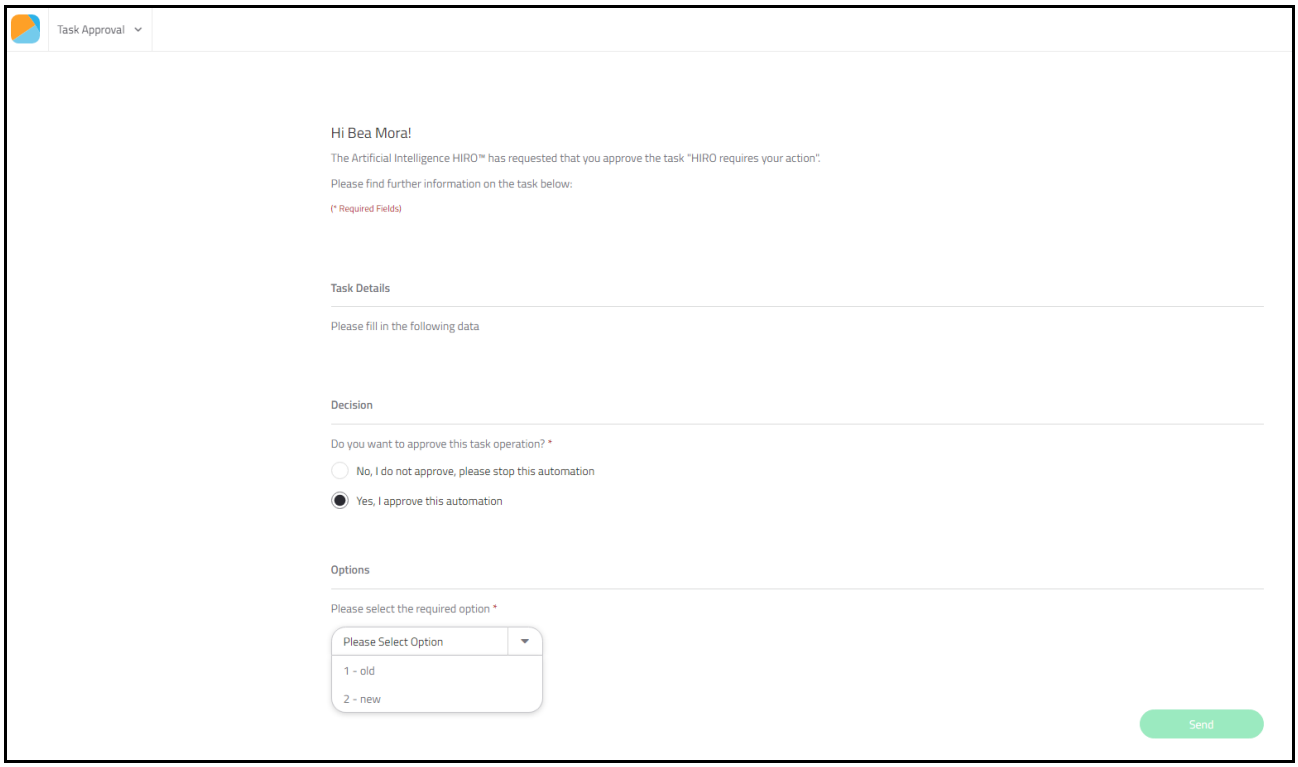
5. MODEL: SELECTING OPTIONS FROM DROPDOWN MENU

```
LOCAL::model = '{
  "action":"approve",
  "task":"${approvalTitle}",
  "template":"${approvalBody}",
  "controls":[
      {
        "approve":{"/approve":"true"},
        "reject":{"/approve":"false"}
      },
      {
        "options": {
         "1 - old": { "/customer": "old" },
         "2 - new": { "/customer": "new" }
       }
      }
    ]
  }'
```


IT's that simple.

In this scenario, it's necessary to incorporate a new parameter called "options." This parameter allows for the creation of multiple options as required. Each option is formatted in JSON with the left-hand side representing the option displayed in the drop-down menu, and the right-hand side containing the variable name along with its corresponding value for each option. Each option is listed on separate lines.

- `"options"`: Holds an object, containing the key (“/customer”), dropdown option names (“1 - old” , “2 - new”), and values (“old”,”new”), to be saved to the DataObject upon selection. There can be multiple keys and values associated with the option block.



You can now view the variables "approve" and "customer" within the Task Content Display, along with their respective values, which can be accessed and utilized by other KIs.



INPUT HEADER AND DESCRIPTION FOR DROPDOWN CONTROL

Just like with the Text control, you can customize the header and description for the DropDown control.

```
{
  "options": {
   "1 - old": { "/customer": "old" },
   "2 - new": { "/customer": "new" }
    },
  "header": "Procedure",
  "desc": "Choose between the procedure 1 or 2"
 }
```

This results in the following layout:

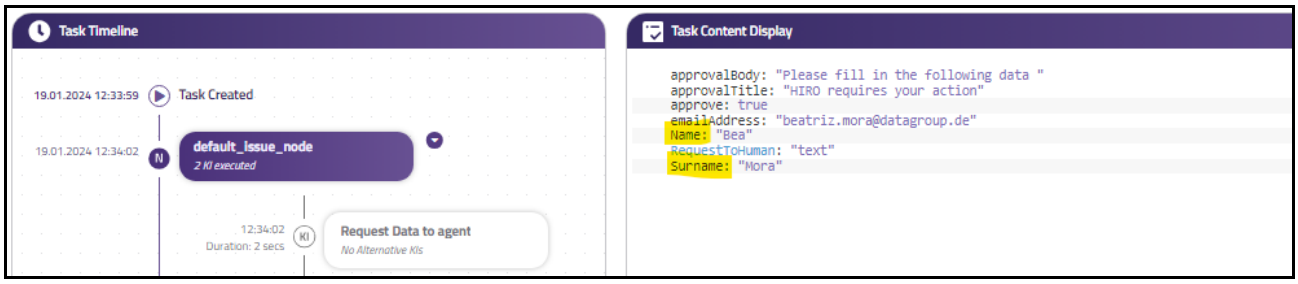


- `"header"`: Optional Header text for this section.
- `"desc"`: Optional Description text for this section.

IT's that simple.

3. TASK CONTENT DISPLAY

In the ¨Task content Display¨ (within Task inspector app) you will observe the newly added variables (Name and Surname) provided by the approver. Additionally, the decision for approval will be reflected in the "approve" variable, providing a comprehensive overview of the completed approval process.



4. AFTER APPROVAL, THE NEXT STEP INVOLVES

You can utilize variables names and values obtained during the approval process within subsequent Kis to initiate further actions, or directly trigger actions via control buttons.

CASE 1

We utilize the "GetApproval" ToDo variable for this step, ensuring its persistence until we reach the decision KI where we validate the value of the "approve" variable. This validation enables us to determine the subsequent actions to be executed.

```
ki
  name: "Get Manager approval”
  description: "Get human approval before executing an action"
  version: "1@cln8sd29t04zz0186ataggjya_clnek6ibg05wx0106o0xre4iv"
on
ogit/name == "default_issue_node"

when
GetApproval == "choice"
ogit/_id # ogit/_id from the task
approvalTitle
approvalBody
emailAddress

do
#Approval Model
LOCAL::model = '{
  "action":"approve",
  "task":"${approvalTitle}",
  "template":"${approvalBody}",
  "controls":[
      {
        "approve":{"/approve":"true"},
        "reject":{"/approve":"false"}
      },
      {
        "text": ["/Name", "/Surname"]
      }
    ]
  }'

exec: LOCAL::RESULT,
  stdout: LOCAL::OUTPUT,
  stderr: LOCAL::ERROR,
  exit: LOCAL::SYSTEMRC = action(
      capability: "AskApprove",
      user_email: emailAddress,
      issue_id: ISSUE::ogit/_id,
      question_model: LOCAL::model
      )
  log.info(LOCAL::RESULT)
  log.info(LOCAL::OUTPUT)
  log.error(LOCAL::ERROR)

  if LOCAL::SYSTEMRC == 0 then
    log("SUCCESS: ${LOCAL::OUTPUT}")
  else
    log.error("ERROR ${LOCAL::ERROR}")
  end

  eject_issue(message: "Wait for approve", issue_id: ISSUE::ogit/_id)
```

```
ki
  name: "Approved received, add member"
  description: "Approved received, add member"
  version: "1@cln8sd29t04zz0186ataggjya_clnek6ibg05wx0106o0xre4iv"
on
ogit/name == "default_issue_node"

when
GetApproval
approve == "true"

do
log("Task continues in HIRO")
AddMember = "true"
delete(GetApproval)
```

CASE 2

Upon approval of the request by the approver, it will automatically trigger the ToDo "AddMember" without the need to add a decision KI.

IT's that simple.

```
ki
  name: "Get Manager approval”
  description: "Get human approval before executing an action"
  version: "1@cln8sd29t04zz0186ataggjya_clnek6ibg05wx0106o0xre4iv"
on
ogit/name == "default_issue_node"

when
GetApproval == "choice"
ogit/_id # ogit/_id from the task
approvalTitle
approvalBody
emailAddress

do
#Approval Model
LOCAL::model = '{
  "action":"approve",
  "task":"${approvalTitle}",
  "template":"${approvalBody}",
  "controls":[
      {
        "approve":{"/AddMember":"true"},
        "reject":{"/approve":"false"}
      },
      {
        "text": ["/Name", "/Surname"]
      }
    ]
  }'

exec: LOCAL::RESULT,
  stdout: LOCAL::OUTPUT,
  stderr: LOCAL::ERROR,
  exit: LOCAL::SYSTEMRC = action(
      capability: "AskApprove",
      user_email: emailAddress,
      issue_id: ISSUE::ogit/_id,
      question_model: LOCAL::model
      )
  log.info(LOCAL::RESULT)
  log.info(LOCAL::OUTPUT)
  log.error(LOCAL::ERROR)

  if LOCAL::SYSTEMRC == 0 then
    log("SUCCESS: ${LOCAL::OUTPUT}")
    delete(GetApproval)
  else
    log.error("ERROR ${LOCAL::ERROR}")
  end

  eject_issue(message: "Wait for approve", issue_id: ISSUE::ogit/_id)
```