Token Management

As a process instance executes, it can be thought of as having a current step that is being processed. If we were to draw out our process on a sheet of paper, we could imagine placing a pebble on the current step being executed. We would think of this pebble as a "token" to indicate which step has control. The IBPM product exposes the concept of tokens to indicate where within the process the current step may be found. The REST API that is exposed by the BPM product provides a set of commands that can be used to interrogate the process to find the current token, a command to move a token from one step to another and a command to delete a token from a process instance. When we combine these capabilities we have the notion of being able to achieve some interesting effects.

By moving a token from one step to another, we effectively have fine grained control over the flow of the process. Specifically, we can jump backwards or forwards within the process. Using this technique we can effectively jump to a previous step and have execution continue from there. Alternatively, we can jump forward skipping current steps and end up at some future step.

Before we go one to discuss these ideas further, let us introduce a note of caution. A process instance is characterized by a number of items. Most importantly of these are the states of variables contained within the process. Steps within a process change or modify the values of these variables. The output of a previous step may be the population of a variable which is essential for the correct operation of a subsequent step. What this means is that if we move a token forward, steps which would have been responsible for variable population are skipped with the result that the variables may not be properly set. When a step is reached which relies on these values, the step may fail to operate as we expected.

In addition, jumping around within a process subverts the design intent of the process designer. A process is a sequence of activities that when executed in order, achieve a well defined business outcome. Jumping around within a process effectively breaks that design. This could result in technical failure of the process or, worse, it could result in erroneous outcomes. For example, if we re-run a step which bills a customer's credit card, the result would indeed be a double billing. If we skip over a billing step, then the customer may not be billed at all. Neither are acceptable outcomes for the process as a whole.

With these notions in mind, the idea of moving a token within a process has to be carefully considered. It should rarely be considered a "go-to" practice and should only be used by expert process designers and even then with caution.

With those disclaimers and caveats in place, let us now look at how this technology can be used.

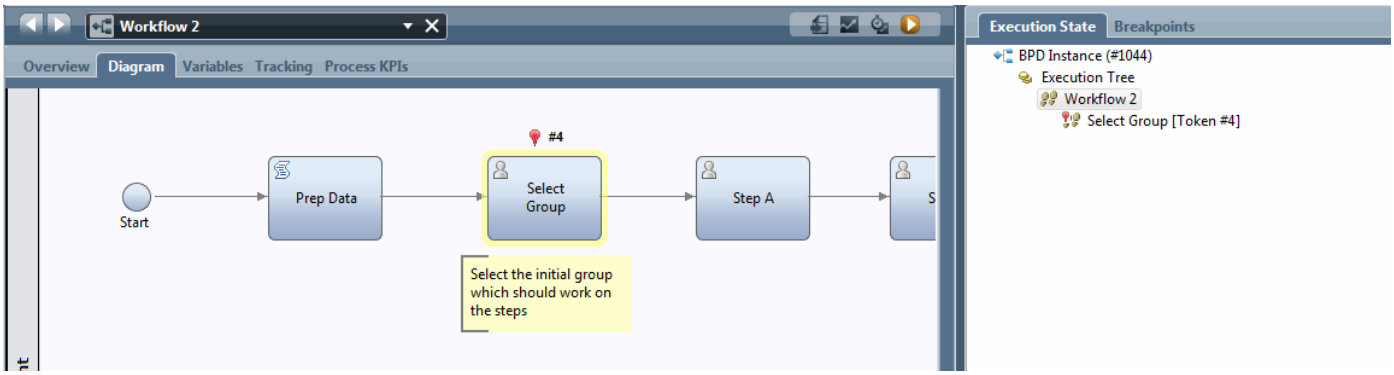
At the core there is a REST API called "`Move Token`" which has the following syntax:

```
POST
/rest/bpm/wle/v1/process/{processInstanceId}?action=moveToken&tokenId={tokenId}&target={stepId}
```

This is quite a technical looking command so let us break it down.

In effect, it is a function call that takes three parameters:

When we run a process within Process Designer and look at it within the Inspector view, we see diagrams that look as follows:



Notice the marker. This is a visual indication of the token within the process. Each token has an identifier associated with it that we all the tokenId. It should be noted that this is not a computable nor fixed value. When a token is at one step it will have a certain tokenId value but as soon as it moves to another step, the token will have a completely new tokenId. It appears in fact to be an illusion that we are "moving tokens". Instead a token is the marker showing which step we are at and when we say we are moving a token, what we really appear to be doing is "ending" the current token and creating a "new"token (with a new tokenId) at the next step. Despite this subtlety, it appears that we can simply think of the token as having moved and still hold our model cleanly. Just ensure that the you realize that a "token" doesn't have the same identifier throughout its life.

How then can we find the current token within a process instance?

Again, another REST command comes to our aid.

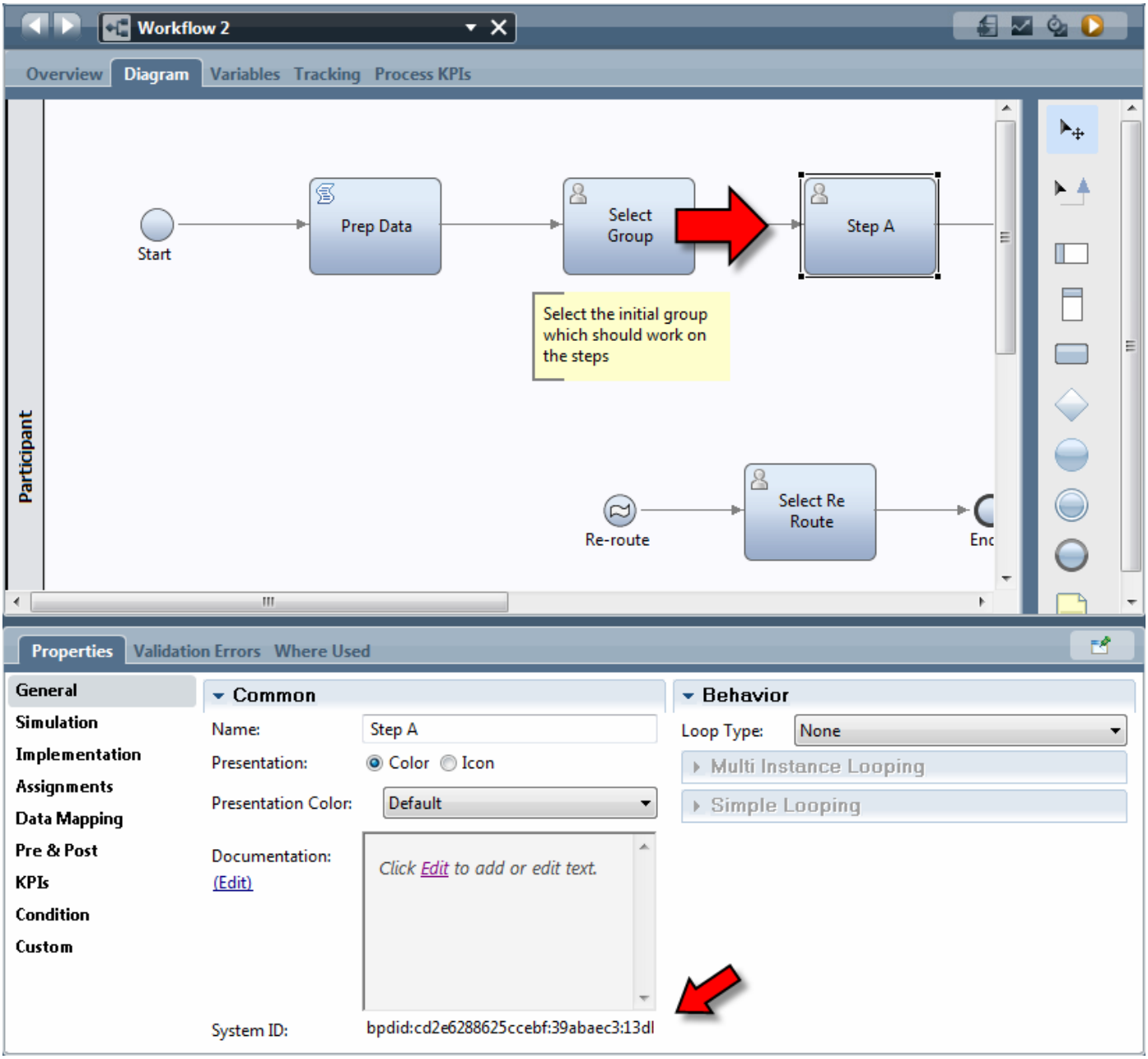
```
GET /rest/bpm/wle/v1/process/{processInstanceId}
```

This function returns a detailed data structure describing the process supplied by the `processInstanceId`. Contained within this data structure is a sub-structure called the "`executionTree`". This structure lists all the steps within the process instance which currently have tokens associated with them. The details of the structure will not be described here and can be found elsewhere. However, the astute reader will have noticed that I attempted to sneak something into our story. I suddenly started to talk about token existence within a process in plural form.

Yes … this is correct and was not a mistaken. Within any given process instance, multiple steps within it may indeed be available to be executed concurrently and hence each of those steps have their own associated token. Because of this, when we are working with process instances and are looking for the identity of a particular token to move, we need to be cautious that a process instance may have multiple tokens and not be fooled into simply taking the first token we come across.

Let us recap. In order for us to move a token within a process, we need to know the `processInstanceId` of the process instance that is going to be manipulated. Knowing the `processInstanceId` we can then ask for the `tokenId` for the token that we wish to move. What remains is to define the step within the process which should be the target of the move. At the conclusion of the move, the token will now reside at that step.

When we look at a process diagram, we see that steps have nice human-readable labels associated with them. In principle, we could use those labels as the identity of a step but the reality is that is a bad idea. If we coupled our logic for moving processes to what a designer chose to name a step on a particular day, we will quickly become broken if someone should simply rename the step. Putting it simply, we don't want the process designers to break our solution if they change a step name nor should we wish to constrain them from changing those names. What we need is some unique identity for a step in a process that isn't going to be related to its label. Fortunately, there is exactly such a thing. If we look in Process Designer at a particular step, we see that in its General Properties tab, there is an entry called "`System ID`". This weird and wonderful code is assured to be unique and distinguishes this step in this process model from any other step in any other process model. Note that to actually see this field in the tooling, you have to enable the "advanced options" within the tool.



With this notion in mind, we can return to the last parameter of our Move Token function. The last parameter which is the target of the token move is exactly an instance of this kind of identifier for a step. Note that if we wish, there is another REST API that will allow us to list all the steps in a process model. This list contains their names as well as their stepIds. From this information, we can present the user a list of possible targets instead of hard-coding.

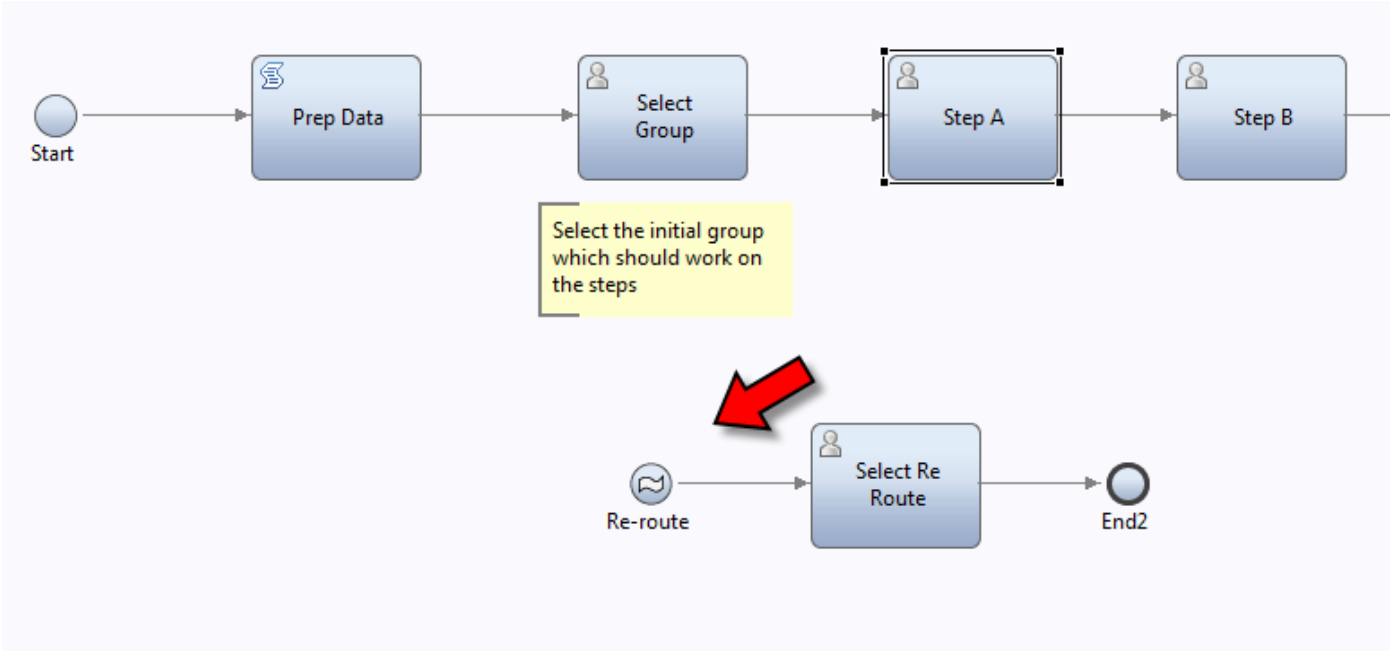
Now we have all the piece parts we need to achieve the token move functions.

Let us now turn our attention to the semantics of moving tokens. If we move a token from a human activity, that human activity is canceled. If we move a token to an activity, it is as though we had just reached that activity through normal flow means and it is executed appropriately.

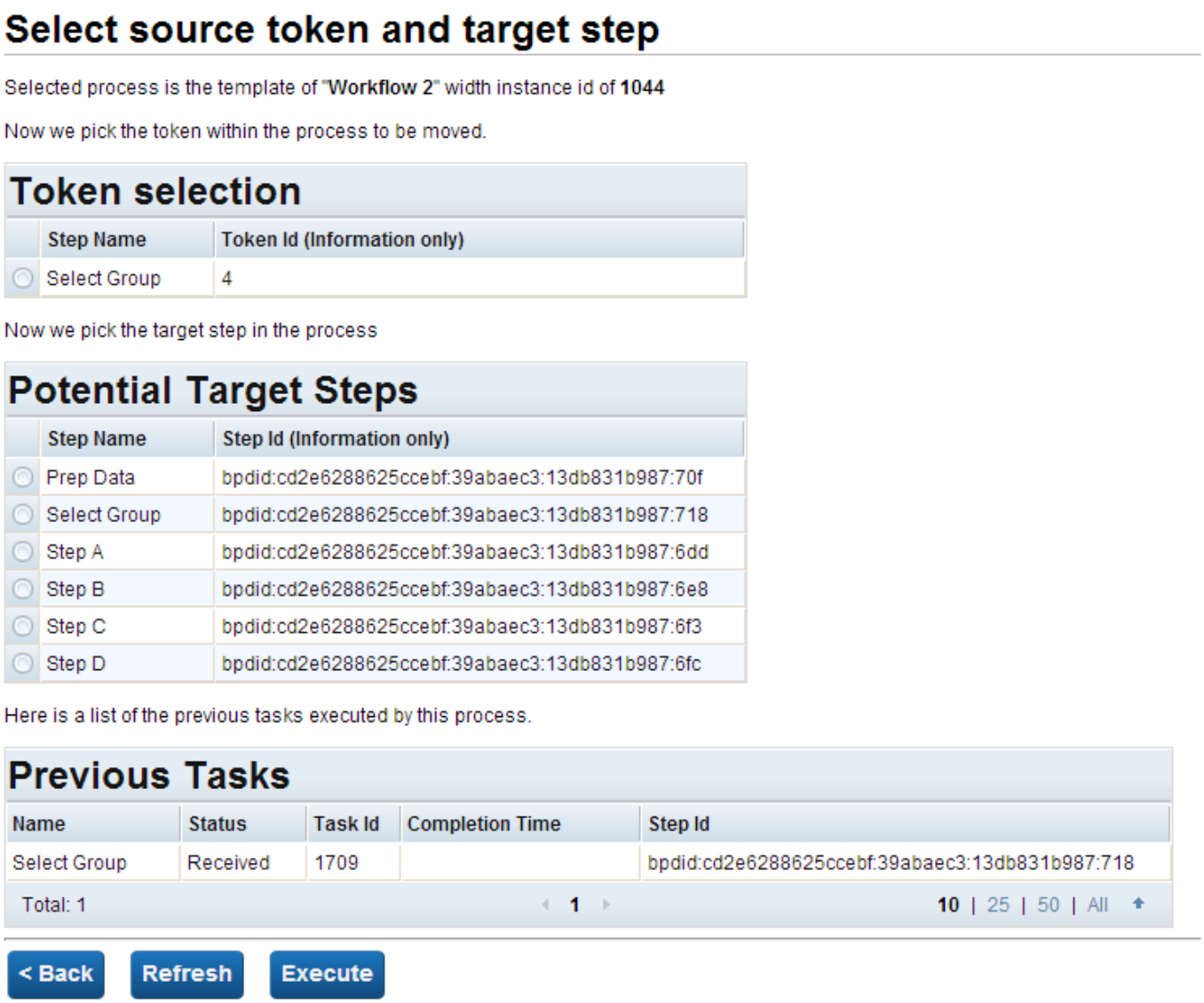
Having now given a brief overview of how to move tokens within a process, let us now look at when and why we would use such a technique.

When we build processes, there are a variety of patterns that we would like to accommodate. Once such pattern is that we may wish to correct an in-flight process instance. Perhaps we have executed a sequence of steps only to realize that a change has been made in our assumptions. Perhaps a customer has called in to amend an existing order or an application for a mortgage has to be reworked because the individual's circumstances have changed. We could always abandon the process and start it again but in certain cases, that may not be the best result.

Within a process we can model an "ad-hoc" entry point. This means that while the process is running, we also have the ability to asynchronously execute steps within its context as a result of an external event. Here we see a process which has an ad-hoc entry point.



When this entry is called, a human service is presented to the user. Within that human service we can design the logic to choose the current token within the process as well as ask choose the step in the process that the token will be moved to. Example screens for this might look like:



Here are shown the list of current tokens and the set of potential target steps. Selecting these appropriately and clicking execute will move the process accordingly.

Revision #1

Created 4 years ago by Admin

Updated 4 years ago by Admin