CloupiaScript Interpreter

About CloupiaScript Interpreter

The CloupiaScript interpreter is a JavaScript interpreter populated with built-in libraries and APIs. You can use the CloupiaScript interpreter to test CloupiaScript code without having to create and run a workflow task.

The CloupiaScript interpreter offers the following built-in functions:

- `PrintObj()` —Takes an object as an argument and prints out all the properties and methods in the object. The printed result provides the names and values for variables in the object and the names of all the object's functions. You can then call `toString()` on any of the method names to examine the method signature.
- `Upload()` — Takes a filename as an argument and uploads the file's contents to the CloupiaScript interpreter.

Starting the CloupiaScript Interpreter

To open the CloupiaScript interpreter, do the following:

Step 1 On the menu bar, choose **Policies** > **Orchestration**.

Step 2 Click the **Custom Workflow Tasks** tab.

Step 3 Click **Launch Interpreter**.

The **Cloupia Script Interpreter** dialog box appears.

Step 4 Click in the text input field at the bottom of the interpreter dialog.

Step 5 Enter a line of JavaScript code and press **Enter**.
The code is executed and the result is displayed. If there is a syntax error in the code, the error is displayed.

Starting the CloupiaScript Interpreter with a Context

You can evaluate JavaScript in the context of a particular a custom task. To do so, you select a custom task and launch the CloupiaScript Interpreter with all the context variables that are defined for executing that custom task.

When you launch the interpreter, the interpreter prompts you for values of the custom task's input fields and populates the input object of the task. All the variables that would be available if you were actually executing the custom task are made available.

To open the CloupiaScript interpreter with a context available, do the following:

Step 1 On the menu bar, choose **Policies** > **Orchestration**.

Step 2 Click the **Custom Workflow Tasks** tab.

Step 3 Choose a custom task for which you need to test the JavaScript.

Step 4 Click the **Launch Interpreter with Context** action.
The **Launch Interpreter** dialog box appears with input fields to collect input values for the custom task. The input fields are those defined for the custom task you have selected.

Step 5 Enter input values in the form.

Step 6 Click **Submit**.

Step 7 Click **Submit**.
The **Cloupia Script Interpreter** dialog box appears.

Step 8 Click in the text input field at the bottom of the interpreter dialog.

Step 9 Enter a line of JavaScript code and press **Enter**.
The code is executed and the result is displayed. If there is any syntax error in the code, the error is displayed.

Example: Using the CloupiaScript Interpreter

The printObj function prints all the properties and methods it contains. You can call the `functiontoString()` to find more details about a function. The following example shows how to examine the ReportContext class and get details about `ReportContext.setCloudName()`.

```
session started
> importPackage(com.cloupia.model.cIM);
> var ctx = new ReportContext();
> printObj(ctx);
```

```
properties =
cloudName:null
class:class com.cloupia.model.cIM.ReportContext
filterId:null
id:null
targetCuicId:null
type:0
ids:[Ljava.lang.String;@4de27bc5
methods =
setIds
jdoReplaceField
jdoReplaceFields
toString
getCloudName
wait
getClass
jdoReplaceFlags
hashCode
jdoNewInstance
jdoReplaceStateManager
jdoIsDetached
notify
jdoGetVersion
jdoProvideField
jdoCopyFields
jdoGetObjectId
jdoGetPersistenceManager
jdoCopyKeyFieldsToObjectId
jdoGetTransactionalObjectId
getType
getFilterId
setType
jdoIsPersistent
equals
setCloudName
jdoNewObjectIdInstance
jdoIsDeleted
getTargetCuicId
setId
setFilterId
jdoProvideFields
jdoMakeDirty
jdoIsNew
requiresCloudName
getIds
notifyAll
jdoIsTransactional
getId
jdoReplaceDetachedState
jdoIsDirty
setTargetCuicId
jdoCopyKeyFieldsFromObjectId

> var func = ctx.setCloudName;
> func
void setCloudName(java.lang.String)
> func.toString();
function setCloudName() {/*
void setCloudName(java.lang.String)
*/}
```