REST API CONTEXT

Bonitasoft

July 2017

SOMMAIRE

1. Objective

The objective of this document is to present the Rest Api Context function.

On Bonitasoft UI Designer, you access the information using a REST API. This is for example the way to access information from the new UI Designer form mechanism, of for a Custom Page.

The objective of this project is to give:

- A simple access: in the same REST call, access all you need in one call. Process Variables, Activity variable, Business Data Object. In the way to want to access it, whatever it's store (variable may be a String, or a Complex java Object)
- A secure access: the user should see only what he has access to see. Some variable, some attributes on a business object, or some business object may be hide for a user. Data access should be control at a process level, or in the BDM level

The RestApiContext is a set of Rest API develop to replace the default set of Rest Api. It must be simple, secure, complete.

2. Access in a process (case, task, overview)

2.1. Principal

The Rest API Context extension access the information with one parameters: the caseId, the taskId, the processinstanceId or the storageContentid. Then using this information, all process variables, all tasks variables, all BDM can be accessible, even if the case or the task is archived (the Standard RestAPI are different for a Archive information and a Active information). So, the RestApiContext can access the information whatever it is.

Nota 1: using the REST API, you have 2 API to access a process Variable (active / archived case), 2 API to access an activity variable (active/archived case), 2 API to access a BDM (active/archived case). So this REST API save you 6 call in one. And if you want to fetch 7 variables, you are supposed to call 7 REST API (14 in an overview form, if you want to display the information for an active and for an archived case).

Nota 2 : the default BDM REST API load a BDM and the children only where they are marked as "load every time". If not, you are supposed to run another(s) REST API for children. So, if one person update the policy in the BDM, the form does not work anymore. This REST API do that for you: it loads children if you ask for.

2.2. In a task

In the task, use

Variable	Type	Expression
formInput	External API	../API/extension/context?taskId={{Id}}

2.3. Overview

in an overview, use

Variable	Type	Expression
formInput	External API	../API/extension/context?caseId={{Id}}

2.4. Case instantiation

In the case instantiation use

Variable	Type	Expression
formInput	External API	../API/extension/context?processId={{Id}}

2.5. Same form for multiple usage

You can use the same form for multiple usage: in a task, in a case instanciation or in the overview.

The UI Designer can't help you to detect where the form is called: the same parameter "id" is used for different usage. Then in a task, the parameter "id" contains the taskId. In the overview, "id" contains the caseId, and in the forminstanciation, "id" is the processdefinitionid !

As you can imagine, if you give the taskId to the "caseid", the information can't be retrieve.

To avoid that, you must give

1/ give the id on the different parameters like

```
FormInput / External API
../API/extension/context?processId={{Id}}&taskId={{Id}}&caseid={{Id}}
```

And RestApiContext will try to retrieve the information according the different parameters. But it's not enough: a taskId can have one day the same value as the processDefinitionId.

2/ give the URL
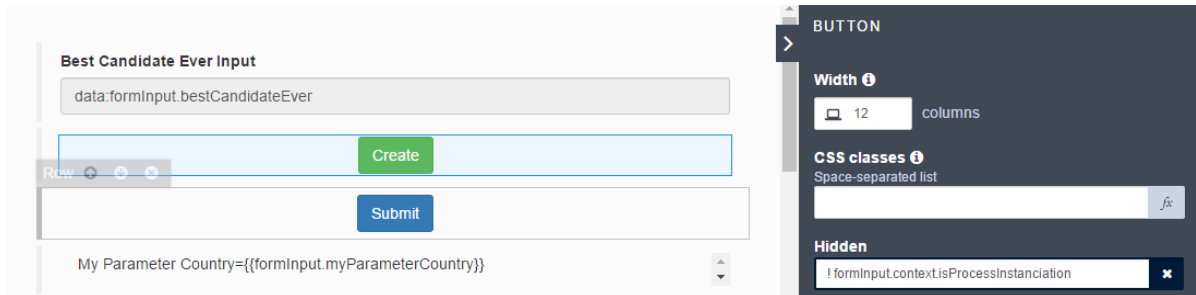
Create a variable "url" as a JAVASCRIPT

Variable	Type	Expression
url	JavaScript	return window.location.href
formInput	External API	../API/extension/context?processId={{Id}}&taskId={{Id}}&caseid={{Id}}&url={{url}}

In the result, in context, the RestAPI context which return in which context you call it: variable isProcessOverview / isTaskExecution / isProcessInstanciation is set.

You can use this variable to display / add the Create button or the Submit button.

CONFIDENTIAL   © 2013 Bonitasoft

In this example, the same form is used for the Process instanciation and for a task. The create is visible only if the condition isProcessInstanciation is not true

Here a complete result of the API:

3. the Pilot

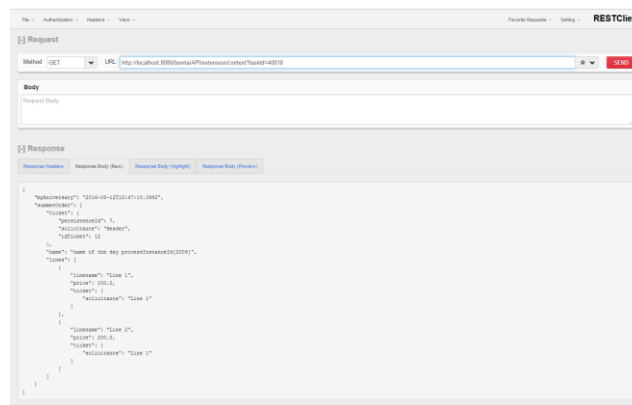By default, the RestApiContext return all variables, parameters and document in the scope of the case.

Via the pilot, it's possible to control what the RestApiContext return.

If you have a security constraints, then you must disable all standard Rest Api, and used only the RestApiContext and pilot to control what the user can receive on each task.

3.6.   In a process

The pilot is a JSON information *saved on the server side* (for security reason), which give the list of data (process variable, activity variable, BDM variables).

The pilot describes the variable who can be accessed, for example:

```
{ "firstname":"public",
  "lastname": "public"}
```

Means the firstname and the lastname variable are public and can be returned. Note that this variable can be some process variable or BDM.

The pilot can be described at multiple level:

- As a local task named "localContext"
- As a global task named "globalContext"
- a parameter named "paramContext".

If no pilot is given, the default one is

```
{ "*":"data"}
```

which mean "load everything"

If you use the RESTAPICONTEXT in an overview, you access in fact the process variable or the parameter variable.

Doing that, you can for example:

Set at the global level a pilot like

```
{ "firstname" : "public" }
```

And in a task "managerreview", you can give the pilot:

```
{ "firstname" : "public", "comment": "public" }
```

Doing that, the variable "comment" is available only in the task "managerreview". To have a complete security, you must disable the Standard Rest API.

You can use a parameter to define the pilot. Main advantage to use a parameter are:

- you can access the value in the formInstanciation (use the URL parameter processId={{processId}} in the REST API)
- you save a variable: a process variable is saved for each case
- you can protect the parameter value. Via a RESTAPI, you can access the parameters value like the database password for example.
- attention: a parameter is accessible by the administrator in the portal, and he can change the value of the parameters

In the pilot, you can reference some complexes information:

```
{
   "invoiceHeader" : {
      "customername" : "public",
      "invoiceid" : "public",
      "invoiceline : {
         "linenumber" : "public",
         "productname" : "public",
         "quantity" : "public",
         "amount" : "public"
      }
   }
}
```

In this example, the "invoiceHeader" can be

- a Java process Variable: first the process variable is transform in JSON and then the result is study to return only the required information.
- The "invoiceHeader" can be a BDM variable too. The invoiceLine is then a another object, referenced by COMPOSITION or AGREGATION, and it can be specified with a LAZY or an EAGER load: the Rest API deal all these constraints to return the correct information, and only the correct information.

If the child is a List of child (in this example, the invoiceLine is a list of elements), then the pilot is applicated on all the different child of the list.

3.7. In a page

This article is only a specification at this moment. The REST API CONTEXT is working from a caseId, a processDefinitionId or a storageId.

In a page, there are no caseId or taskId. That, where the pilot can be found? What is the content of the pilot?

4. User Access

Some control can be given in the REST API access, in order to check the data. In a task, this is in general not a big deal (the REST API CONTEXT first check if the user can access to the tasks), but in a Overview Access, you may want to control the information according who access it.

For example, a case is created by a student, and teacher access the case to give some recommendation. You want then in the variable protect the "teacherComment" : only teacher can see this process variable.

You can set the teacherComment in the task "teacherReview" and not in the task "studentAdditionalInformation":

Localcontext in "studentAdditionalInformation"

```
{
    "studentRequest":"data",
    "teacherDecision" : "data"
}
```

Localcontext in TeacherReview

```
{
    "studentRequest":"data",
    "teacherComment" : "data",
    "teacherDecision" : "data"
}
```

And the globalContext has to be set too (else, the default pilot is "*":"*")

GlobalContext or ParamContext

```
{
    "studentRequest":"data",
    "teacherDecision" : "data"
}
```

Nota: you don't really need the localContext in "studentAdditionalInformation" because the global one is the same.

But, if you want now create an Case Overview, and let's the teacher see it's comment ? here come the userAccess.

In the user access, you can pilot with different value : "initiator" is the user who create the case, actor an actor of the case. You can add multiple access, separate by a semicolon.

The GlobalContext or ParamContext maybe:

```
{
    "studentRequest":"data",
    "teacherComment" : "actor:teacher,task:teacherReview ",
    "teacherDecision" : "data"
}
```

With this user access, all users registered in the actor "teacher" OR all user who executed the task "teacherReview", will have the attribute "teacherComment" in the result.

Note: in the current version 2.7, the userAccess works only on process variable, not on BDM

This following is only a specification at this moment.

In order to give the access from a Custom Page, it's possible to declare a "accessrigth"

```
{
    "myAniversary":"data",
    "summerOrder" : {
       "name": "data",
       "ticket": "*",
       "lines" : {
              "linename" : "data",
              "ticket" : { "solicitante" : "data" },
           "price":"data"
       }
    }
}
```

It's possible to declared pilot and access rule

```
{
  "pilotname" [
    "public" : {
      "myAniversary":"data",
      "summerOrder" : {
        "name": "data",
        "ticket": "*",
        "lines" : {
          "linename" : "data",
          "ticket" : { "solicitante" : "data" },
            "price":"data"
          }
        }
      }, // end public
      "manager" : {
        "myAniversary":"data",
        "comment":"data",
        "summerOrder" : {
          "name": "data",
          "acceptance": "data"
        }
      } // end manager
    } // end public
  ],
  "accessrigth": [
      {"name": "To access the manager pilot",
       "access" : [ "Profile|administrator", "Group|/Acme/hr", "User|Helen.Kelly" ],
       "pilot" : "manager"
      },
      { "name": "Public"
        "access" : [],
        "pilot":"public"
      }
  ]
}
```

4.8.   Control

The control is defined in the Pilot as the value of the attributes. The control contains the Access Rigth (see before) and the format part.

An example of pilot is:

```
{
    "firstname" : "data",
    "comment": "initator;task:review",
    "adresse" : {
        "street" : "actor:Verify Adresse;admin",
        "city" : "actor:Verify Adresse",
      },
    "ticket" : "*"

}
```

Nota: when the value is a Date, it can be formatted using the explicite format. Else, the default value can be set in a Configuration File.

4.8.1. Data or public

If the control is "data" or "public", no special operation is done.

4.8.2. Initiator

Only the initiator user (the user who create the case) can access this attribute.

4.8.3. Admin

User is register in the administrator profile.

4.8.4. Supervisor

User is register in the process manager.

Supervisor and sub-process

In the situation where

- A sub-process exists
- A case 4453 is created, and a task 180322 is pending in the sub-process
- A permission "supervisor" is set on the data "comment".

When you access the REST API by the taskId, then the taskId is found in the sub-process, and the supervisor is verified in the sub process; If the user is registered as a supervisor, he can view the data

When you access the REST API by the caseId, then there is no information on the sub-process (and maybe due to a gateway, you may have multiple task in multiple sub-process). So, the supervisor is done at the Parent process.

The best in that situation is to register the user as a supervisor in both process

4.8.5. Task

Only person who are candidates on the task or who executed the task, can access this attribute

4.8.6. Actor

Only person registered in this actor can access this attributes.

Actor and sub-process

In the situation where

CONFIDENTIAL   © 2013 Bonitasoft

- A sub-process exists
- A case 4453 is created, and a task 180322 is pending in the sub-process
- A permission "actor:reviewer" is set on the data "comment".

When you access the REST API by the taskId, then the taskId is found in the sub-process, and the actor 'reviewer' is verify in the sub process; If the user is registered in this actor, he can view the data

When you access the REST API by the caseId, then there is no information on the sub-process (and maybe due to a gateway, you may have multiple task in multiple sub-process). So, the actor verification is done at the Parent process.

The best in that situation is to define the same actor (same name) in the Parent process.

4.8.7. Format:date

The value is supposed to be a date, and then the date will be returned in the format "yyyy-MM-dd"

4.8.8. Format:datetime

The value is supposed to be a date, and then the date will be returned in the format "yyyy-MM-dd'T'HH:mm:ssZ"

4.8.9. format:datelong

The value is supposed to be a date, and the value will be return as a time stamp (JAVA : myDate.getTime()).

4.9. Access Right

You can use an access right to simplify and organize the access. Access right is useful to change the access according the status of the case: via the access right mechanism, you can change the access when the case is archived.

The usage of the access is the following:

```
accessright:<name>
```
For example:

```
{
   "firstname" : "accessright:student",
   "lastname"  : "initator;accessright:student;accessright:teacher ",
}
```
To define the access right, the pilot variable execution context ("__execution").

Different accessright perimeter can be used:

perimeter	explanation
accessright	Default one
accessrightinstantiation	Use at the instantiation
accessrightoverview	Use at the overview
accessrighttask	Use at a task
accessrightarchive	Use when the case is archive (overview case then)
accessrightcase	Use when an active case is access

For example, by this definition, the student access right is defined:

```
{
    "__execution" : {
        "accessright" : {
            "student" : "role:isStudent;initiator",
            "teacher" : "role:isTeacher;group:/acme/hr"
        },
        "accessrightoverview" : {
            "student" : "role:isStudent;initiator",
            "teacher" : "role:isTeacher;group:/acme/hr;role:isManager"
        },
    }
    "firstname" : "accessright:student",
    "lastname"  : "initator;accessright:student;accessright:teacher ",
}
```

Then, during the case overview, a user which part of the role "isManager" can see the lastName. If the same user can access a task, this information is not visible.

5. Explicit variable

The RestApi context contains a mechanism to let the configuration calculate explicit variable.

The first behavior of the RestApiContext is to access some case variable; As example, the process contains a variable "application" (BDM variable or process variable). Then, by the pilot

```
{
    "application" : "public ",
}
```

The variable is returned.

By an explicit variable, the way to collect the information may change. As an example, the process can call a iterator and a sub process. Then, each sub process will change part of the application variable. During this part of the process, the "application" variable accessible at the parent process is not up to date. And due to the iterator and the call activity, there are one "application" per sub process / iterator, and all the different sub process variable should be aggregate to have a photo of "application".

Via the explicit variable, it's possible to integrate this behavior. Via this mechanism, it's possible too to return some data which are not part of the process. And the explicit variable can be use to give a default value, at the instantiation for example.

The explicit variable is a "first floor". When the variable is collected, then the pilot is used to decide which part of the variable must be send back in the result, according the pilot definition and access right.

It's possible too to define a different explicit variable for a default access, or for a case overview, or for an archived case.

perimeter	explanation
explicit	Default one
explicitinstantiation	Use at the instantiation
explicitoverview	Use at the overview
explicittask	Use at a task
explicitarchive	Use when the case is archive (overview case then)
explicitcase	Use when an active case is access

The definition must be done in the execution context variable ("_execution"). Here an example,

```
{
   "__execution" : {
      "explicitoverview" : {
         "application": "java:org.bonitasoft.application.Load({{caseid}}#myDataSource)"
      },
      "explicitinstantation" : {
         "durations": "json:{ \\"DLUO\\": 12, \\"conso\\": 5}"
      }
   },
   "application" : "public",
   "durations" : "public"
}
```

The different order acceptable are:

Explicit order	explanation
json	Give a JSON information
Java	Call a JAVA method

The JAVA method must respect this signature:

```
Public static Object myName( string parameter)
```
And return an object.

The parameters is calculated from the definition. The following place order are available:

Place holder	explanation
{{processid}}	The process definition id.
{{caseid}}	The case id. In case of an archive case, the original case Id is given
{{taskid}}	The tasked. If the access is during a case overview, then this place holder return ""

So, theses parameters are valid

```
java:org.bonitasoft.application.Load({{caseid}}#myDataSource)
java:org.bonitasoft.application.getMyData(Hello my {{caseid}} from a
{{processed}}myDataSource)
```
the object must accessible from the RestApiContext.

The best way is to deploy the JAR file directly in the RESTCONTEXT:

- Open the ZIP file RestContext.zip
- Add the JAR file in the lib directory
- Deploy it.

6. Dates

The dates management are complex.

There are different ways to manage dates:

- Using the Community Widgets dates or the Standard widget dates
- Using the java Date type or the JDK 1.8 java.time.localDate

You can pilot the way the RestApiContext return a result by the DateFormat configuration. This DateFormat can be give at the URL:

http://localhost:8080/bonita/portal/resource/taskInstance/dates75/1.0/View%20All%20My%20Date/API/extension/context?taskId=20003&processId=20003&dataformat=DATETIME

or in the Configuration properties file

```
# Possible value : DATELONG, DATETIME, DATEJSON
DEFAULT_DATE_FORMAT=DATETIME
```

Note that the different value will work correctly with all widget, so the default one, DATETIME, are in general the best for debugging reason.

The main different is visible when you want to use a java.time.localDate with an "Absolute" widget.

6.10.   Date anniversary

Use case: the user give a date (like Feb 4 2017) and you want to save in the process variable the date Feb 4 2017 00:00:00. This date is the same on all the time zone.

Widget Bonita DatePicker	

Save in java.util.Date	Use TEXT in the contract
	
	Transform with
	``
	SimpleDateFormat sdf;
	sdf = new SimpleDateFormat("yyyy-MM-dd'T'HH:mm:ss");
	
	sdf.parse(aniversaryInput);
	``
	or use DATE ONLY in the contract:
	``
	import java.time.ZoneId;
	Date.from(dateAniversaryInput.atStartOfDay(ZoneId.systemDefault()).toInstant()))
	``

Get RestApiContext	Nothing special ``` { 'aniversaryDate' : 'data' } ```
Save in java.time.localDate	Use DATE ONLY in the contract Map the variable to the contract
Get RestApiContext	Nothing special ``` { 'aniversaryDate' : 'data' } ```

Nota: if you want to use the Contract the type "DATE ONLY" and save in the BDM or in a process Variable a java.util.Date, use the second transformation

6.11.   Date Open Hour

Use case: the user give a date (like Feb 4 2017 10:23:00) and you want to save in the process variable the date Feb 4 2017 10:23:00. This date is the same on all the time zone. This is for example the date + time that a shop open in the word whatever the time zone (it's open a 10:23 at PARIS and at 10:23 at SAN FRANCISCO).

In 7.5 a Bonita widget is available.

Widget Bonita DatePicker handle local time zone: NO	

Save in java.util.Date	Use TEXT in the contract
	
	Transform with
	<pre>SimpleDateFormat sdf;
	sdf = new SimpleDateFormat("yyyy-MM-dd'T'HH:mm:ss");
	
	sdf.parse(aniversaryInput);</pre>or use DATE ONLY (no time zone)
	<pre>import java.time.ZoneId;
	
	return
	Date.from(dateOpenHourInput.atZone(ZoneId.systemDefault()).toInstant());</pre>

Get RestApiContext	Transform the value (1) <pre>{ 'aniversaryDate' : 'format:absolute' }</pre>
Save in java.time.localDateTime	Use DATE TIME (NO TIME ZONE) in the contract Map the variable to the contract
Get RestApiContext	Transform the value (1) <pre>{ 'aniversaryDate' : 'format:absolute' }</pre>

(1) The Bonita Standard widget with a status "no time zone" is not enough to get back the correct value. It must know that the value is absolute too (and then the parameter between the value and the widget must be synchronized, else the result is not correct).

The REST API return then "2017-12-14T10:00:00" with an ABSOLUTE format, and 2017-12-14T10:00:00Z by defaut (the Z is present or not).

With the Widget Community:

Widget Bonita DatePicker mode ABSOLUTE	

Save in java.util.Date	Use TEXT in the contract
	
	Transform with
	<pre>SimpleDateFormat sdf;
	sdf = new SimpleDateFormat("yyyy-MM-dd'T'HH:mm:ss");
	
	sdf.parse(communityOpenHour);</pre>

Get RestApiContext	Nothing special <pre>{ 'aniversaryDate' : 'data' }</pre>
Save in java.time.localDate	Use DATE TIME (NO TIME ZONE) in the contract Map the variable to the contract
Get RestApiContext	Nothing special <pre>{ 'aniversaryDate' : 'data' }</pre>

The Community widget manage the same data, and it's part of its configuration to display the date. In ABSOLUTE mode, the widget just display the same date. In the Time Zone management, the widget will consider the date as a UTC one and will use the browser local time zone to display the date correctly in the local time zone.

6.12.   Date Meeting

Use case: the user give a date (like Feb 4 2017 08:23:00 PDT) and you want to save in the process variable the date at the UTC zone(Feb 4 2017 17:23:00 UTC). The date is then recalculate according the time zone of the brower : user in New York should see the date in its time zone ((Feb 4 2017 11:23:00 EST).

In 7.5 a Bonita widget is available.

Widget Bonita DatePicker handle local time zone: YES	

Save in java.util.Date	Use TEXT in the contract
	
	Transform with
	```\nSimpleDateFormat sdf;\nsdf = new SimpleDateFormat("yyyy-MM-dd'T'HH:mm:ss");\n\nreturn sdf.parse(aniversaryInput);\n```
	or use DATE TIME (TIME ZONE)
	```\nimport java.time.format.DateTimeFormatter;\nimport java.text.SimpleDateFormat;\n\nDateTimeFormatter dtf = DateTimeFormatter.ofPattern("yyyy-MM-dd'T'HH:mm:ss");\nSimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd'T'HH:mm:ss");\n\nReturn sdf.parse(dtf.format(magicMeeting));\n```

Get RestApiContext	Nothing special ```\n{ 'aniversaryDate' : 'data' }\n```
Save in java.time.OffsetDateTime	Use DATE TIME (TIME ZONE) in the contract Map the variable to the contract
Get RestApiContext	Nothing special ```\n{ 'aniversaryDate' : 'data' }\n```

With the Widget Community:

Widget Bonita DatePicker mode ABSOLUTE	

Save in java.util.Date	Use TEXT in the contract
	
	Transform with
	<pre>SimpleDateFormat sdf;
	sdf = new SimpleDateFormat("yyyy-MM-dd'T'HH:mm:ss");
	
	sdf.parse(communityOpenHour);</pre>or use DATE TIME (TIME ZONE)
	
	<pre>import java.time.format.DateTimeFormatter;
	import java.text.SimpleDateFormat;
	
	DateTimeFormatter dtf = DateTimeFormatter.ofPattern("yyyy-MM-dd'T'HH:mm:ss");
	SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd'T'HH:mm:ss");
	
	Return sdf.parse(dtf.format(magicMeeting));</pre>

Get RestApiContext	Nothing special <pre>{ 'meetingDate' : 'data' }</pre>
Save in java.time.OffsetDateTime	Use DATE TIME (TIME ZONE) in the contract Map the variable to the contract
Get RestApiContext	Nothing special <pre>{ 'meetingDate': 'data' }</pre>

7. Context

The REST API return a "context" information. The context contrains multiple information :
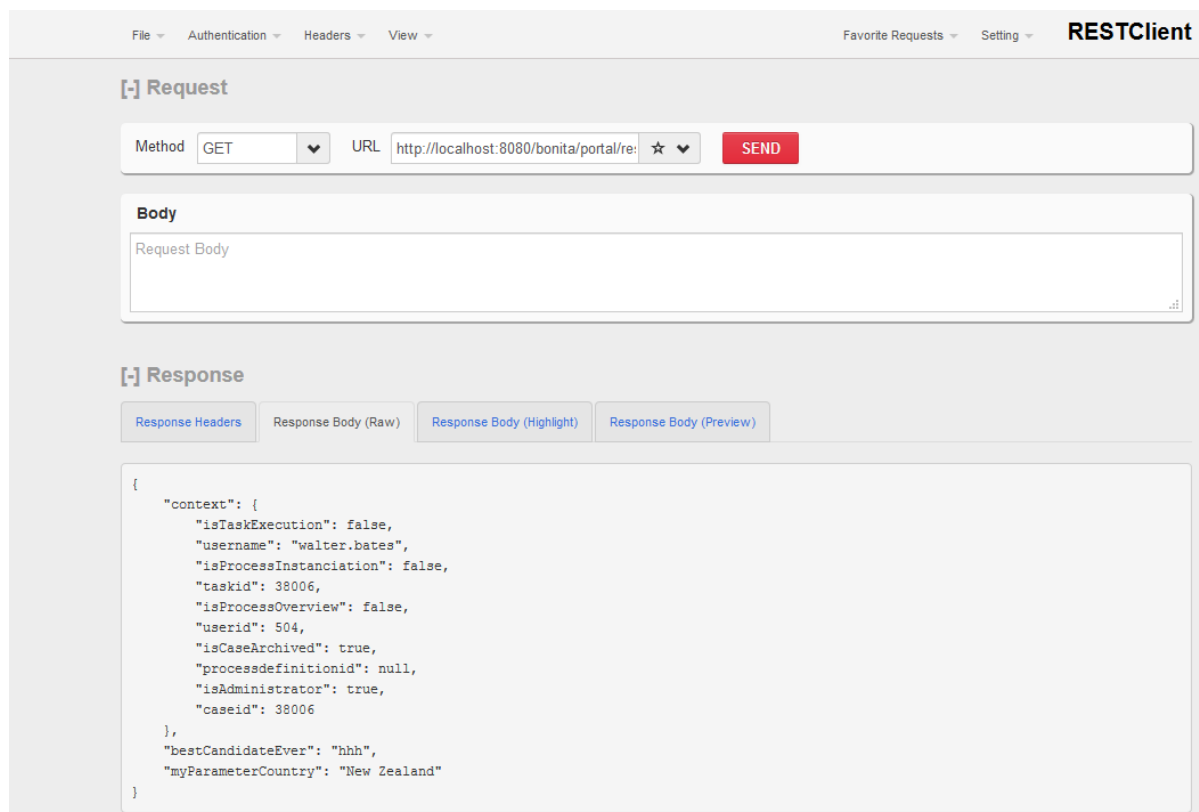
```
{
  context{
    isAdministrator:true,
    isSupervisor:false,
    isProcessInstanciation:true,
    isProcessOverview:false,
    isTaskExecution:false,
    isCaseArchived:false,
    isTaskArchived:false,
    processdefinitionid:8067288333288041000
    caseid:38006,
    taskid:4453,
    taskname: "Check the expense",
    userid:304,
    username:"walter.bates"
  }
```

- isAdministrator : the user is register in the administrator profile
- isSupervisor: the user is register in the Process manager
- IsProcessInstanciation, isProcessOverview, isTaskExecution, isCaseArchived, isTaskArchived : the RestAPI decode the URL and return the context of the execution : in an overview, a process instantiation or a task execution
- caseid, taskid, processdefinitionid : the parameter is fulfill according the execution. In a task, all the information is provided.

- Userid and username: the name and the id of the user executed the rest api.
- Taskname: the name of the name if the request is part of a task
- Caseiduse: the case is maybe in a sub process, then this is the subprocess caseid
- Caseidparent : the case is maybe in a subprocess, then give the parent caseid

If the log is true in the URL, more information is returned:

- allowContext : an explanation why the user can access the context
- performanceDetail : give a detail of the execution time
- performanceTotalMs : the execution time
- sourcecontextdata : explain where the pilot is found

8. Access documents

The document is included in the REST API CONTEXT and security download is provide (upload is managed via the contract).

In a process,
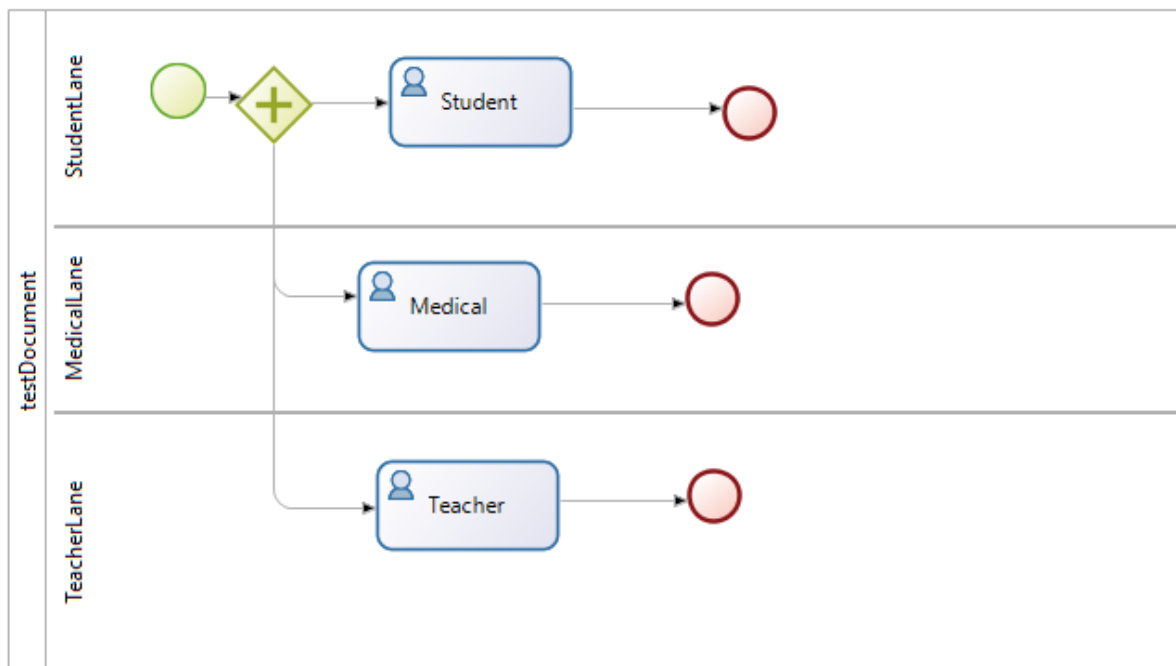
you create 3 documents, "publicdocument", "medicaldocument", "teacherdocument".

8.13. Information in the document

In the context, specify that the document must be in the answer.

On a task, the information about the document respect the "FileUploadPlus" design. If you use a standard Link or a Standard FileViewer, you have to use the <documentname>.src to access the information.

```
publicDocument:{
  src:{
    author:304,
    contentFileName:"aDoc 1.pdf",
    contentMimeType:"application/pdf",
    contentStorageId:"301",
    creationDate:1490727279205,
    description:"",
    fileName:"aDoc 1.pdf",
    hasContent:true,
    id:301,
    index:-1,
    name:"publicDocument",
    processInstanceId:12001,
    url:"documentDownload?fileName=aDoc 1.pdf&contentStorageId=301",
    version:"1"
  }
}
```

Then, use this URL in the LINK widget

'/bonita/portal/'+formInput.publicDocument.src.url

          © 2013 Bonitasoft

8.14.   Permission to access the document

Idea is:

- The document is describe with a control "data" : the document is accessible to be download

```
"publicDocument" : "data"
"publicDocument" : "public"
```

- The document is describing with a control "initiator", only the case creator can access the document

```
"publicDocument" : "initiator"
```

- With a control "task:<name>": if the task is active, only candidate can access it. If the task is archived, only person who executed this task can access it.

```
"medicalDocument" : "task:Medical"
```

- With a control "actor:<actorName>" : only user maps to this actor can access it.

```
"teacherDocument" : "actor:teacherActor"
```

A combination is possible to maps the different element, using the semicolon:

```
"teacherDocument" :"initiator;actor:teacherActor;task:Teacher"
```

Doing that, the information to download the document are presented or not. Not the information follow the FileWidgetUpload requirement.

To ensure the security, a new DownloadContext.groovy is provided. This groovy script replaces the default one (the default one check only the basic security: if you can access the case, you can access the document, which correspond to the control "publicDocument":"data"). This new script will check the security according this rule. The only information given is the "storageId". From the "storageId", the documentId, and then the CaseId can be found.

- Data: if the user can access the case, the download is allowed
- Creator: if the user is the case creator, the download is allowed
- Task: in the case, a search to the task is done. If the task is active, and the user is part of the candidate of the task, download is allowed. If the task is archived, the user must have executed it
- Actor: in the case, a search to the actor member is done. If the user is part of the actor member, the download is allowed.

8.15.   Configure the server

The Document upload is managed by the Dynamic Check. To enable the permission, the installation is

8.15.10.   Setup pull

7.3 and Over : Use the setup tool to upload the configuration file

8.15.11.   DocumentPermissionContextRule file

Put the different on the security path <bonita_home>\engine-server\work\tenants\1\security-scripts:

Unzip the file DocumentPermissionRule.zip

It contains 5 files.

8.15.12. Enable the dynamic

In the file In < bonita_home>\client\tenants\1\conf\security-config.properties, verify that the security is enable:

```
#Setting this value to false will deactivate the permissions checks on the REST API
security.rest.api.authorizations.check.enabled true
```

8.15.13. Dynamics security check:

Enable the Dynamics security check:

In < bonita_home>\client\tenants\1\conf\dynamic-permissions-checks.properties

```
##Servlets
GET|portal/documentDownload=[profile|Administrator, check|DocumentPermissionContextRule]
GET|portal/formsDocumentDownload=[profile|Administrator,
check|DocumentPermissionContextRule]
GET|portal/downloadDocument=[profile|Administrator, check|DocumentPermissionContextRule]
# DocumentPermissionRule
## Let a user access only document on cases that he is involved in
GET|bpm/document=[profile|Administrator, check|DocumentPermissionContextRule]
POST|bpm/document=[profile|Administrator, check|DocumentPermissionContextRule]
PUT|bpm/document=[profile|Administrator, check|DocumentPermissionContextRule]
DELETE|bpm/document=[profile|Administrator, check|DocumentPermissionContextRule]
GET|bpm/archiveddocument=[profile|Administrator, check|DocumentPermissionContextRule]
GET|bpm/archivedCaseDocument=[profile|Administrator, check|DocumentPermissionContextRule]
GET|bpm/caseDocument=[profile|Administrator, check|DocumentPermissionContextRule]
POST|bpm/caseDocument=[profile|Administrator, check|DocumentPermissionContextRule]
PUT|bpm/caseDocument=[profile|Administrator, check|DocumentPermissionContextRule]
DELETE|bpm/caseDocument=[profile|Administrator, check|DocumentPermissionContextRule]
```

8.15.14. Web.xml

In <tomcatHome>/webapp/bonita/WEB-INF/web.xml

Uncomment the url-patter on documentDownload

```
<filter-mapping>
        <filter-name>RestAPIAuthorizationFilter</filter-name>
        <url-pattern>/API/*</url-pattern>
        <url-pattern>/APIToolkit/*</url-pattern>
        <!-- see TokenValidatorFilter comment -->
        <url-pattern>/portal/custom-page/API/*</url-pattern>

        <!-- Uncomment those lines to apply security checks also on portal servlets -->
        <url-pattern>/portal/formsDocumentDownload</url-pattern>
        <url-pattern>/portal/formsDocumentImage</url-pattern>
        <url-pattern>/portal/documentDownload</url-pattern>
        <url-pattern>/portal/downloadDocument</url-pattern>
        <url-pattern>/portal/custom-page/API/formsDocumentImage</url-pattern>
        <url-pattern>/portal/custom-page/API/documentDownload</url-pattern>
        <url-pattern>/API/formsDocumentImage</url-pattern>
        <url-pattern>/API/documentDownload</url-pattern>
```

8.15.15. Setup push

7.3 and over : push the configuration to the server

8.15.16. Restart the server

Then restart the server

                   © 2013 Bonitasoft

9. Configuration

In the REST API CONTEXT, you can find a file configuration.properties. The properties is used as default.

To change it :

- Edit the file and change the value
- Redeploy the REST API EXTENSION

Here the default configuration file

```
# Possible value : DATELONG, DATETIME, DATEJSON
DEFAULT_DATE_FORMAT=DATETIME
```

10. JDK 1.7

The RestApiExtension is built to run under JDK 1.8 (which is the mandatory version for Bonita 7.5 and upper).

To run it on a lower version:

- Install the RestApiContext source (see 12.18 Importing the source code)
- Remove the class RestContextTransformData_18.groovy
- In the pom.xml, change the version 1.8 to 1.7 in the maven-compiler-plugin

```
<groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-compiler-plugin</artifactId>
    <version>3.3</version>
    <configuration>
        <source>1.7</source>
        <target>1.7</target>
        <compilerId>groovy-eclipse-compiler</compilerId>
    </configuration>
```
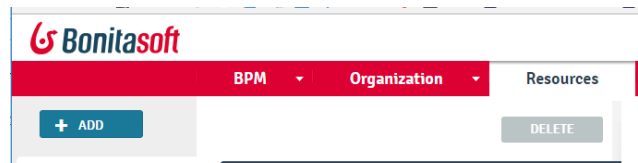
11. Access in a custom page

This article is only a specification at this moment. The REST API CONTEXT is working from a caseId, a processDefinitionId or a storageId.
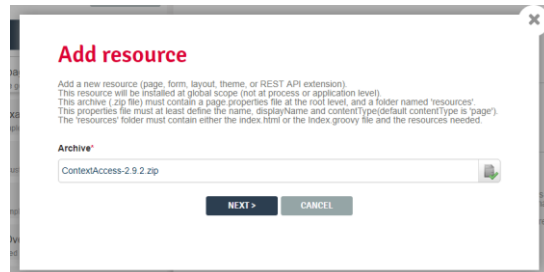
12. Installation

12.16. Deploy RestApiContext

In the Bonita Portal, as and Administrator, go to Resources and click on Add

Then select the ContextAccess zip file:

Click on Next, then Confirm.

## 12.17.   Using DemonstrateRestApiContext

This BOS file contains several process example and a BDM. To use it:

- Be sure that you can overwrite your local BDM (change the repository, or the workspace)
- Load the RestApiContext.zip in the Portal (see previous declaration)
- Import the BOS file DemonstrateRestApiContext
- Deploy the BDM : go to Development/ BDM and the
- Deploy the Groovy libray
 If at execution you get
  Caused by: org.bonitasoft.engine.expression.exception.SExpressionEvaluationException.: Expression getMyAnimal with content = <return PetsAnimals.TURTLE;> depends on PetsAnimals is neither defined in the script nor in dependencies. ( )
  Go to Development /Manage groovy Script and verify that you have AdditionnalDeclaration groovy libs
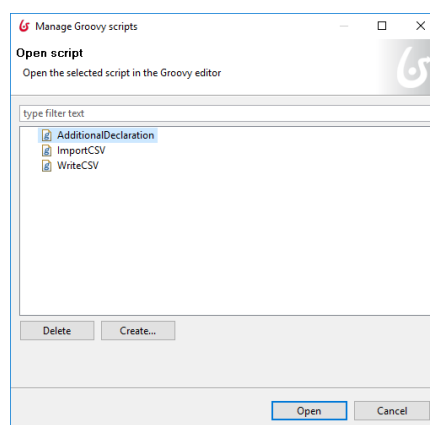
Then go to the process ContextCall, in "Process Dependencies" and check the AdditionalDeclaration groovy script
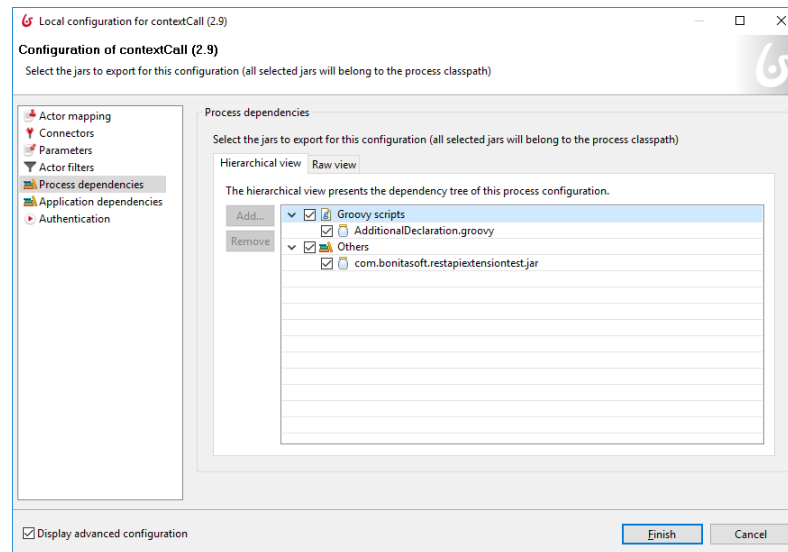
12.18. Importing the source code

To import the Source Code (and then to remove all 1.8 reference)

1. Import source file

Click on Import and select the RestApiContext.bos file

The source code use the BonitaEvent librairy, via the POM:

```
<dependency>
    <groupId>com.bonitasoft.log.event</groupId>
    <artifactId>bonita-event</artifactId>
    <version>1.0.0</version>
</dependency>
```

2. Check the Bonita version

In the pom.com, check the Bonita version and give the correct on:

```
<properties>
    <bonita.version>7.2.4</bonita.version>
```

3. Import BonitaEvent in Maven

To get this library, download it from Github;

Go to https://github.com/Bonitasoft-Community/BonitaEvent/releases and upload the last version

Install it via:

```
mvn -X install:install-file -Dfile=bonita-event-1.0.0.jar -DgroupId=com.bonitasoft.log.event
-DartifactId=bonita-event -Dversion=1.0.0 -Dpackaging=jar
```

4. Import RestContextComponent in Maven

This component is part of the Java aspect of the RestApiContext. When all the repository is fork, this is present under src/main/java.

To just import the missing JAR file, download it from the release, and import it in Maven

```
mvn -X install:install-file -Dfile=bonita-event-1.0.0.jar -DgroupId=com.bonitasoft.log.event
-DartifactId=bonita-event -Dversion=1.0.0 -Dpackaging=jar
```

5. Update the project

On the Studio, select "ContextAccess", and Right click on. On the contextual menu, select "Maven/update project"

1.
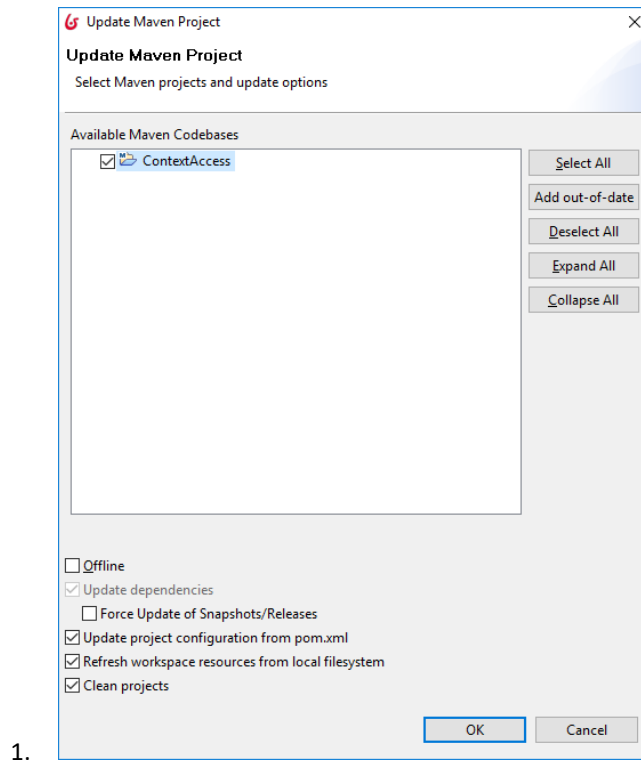
13. Questions

13.19. Where is the Pilot?

The pilot is saved in a parameter, a variable, or a local variable. There are some aspects:

- In a variable or a local variable: each case created, the information is duplicated in the database
- In a variable or a local variable: when you want to change the pilot, you must deploy a new process. And with Bonita Subscription, you can upload a new form without deploying the process
- In a parameter: you must setup the parameter for each environment
- In a parameter: protect correctly the administration page, because with a subscription, administrator can change the parameter/

And on a Custom page, there are no process, so the pilot is not accessible.

So, we want:

- to be able to describe the pilot process per process, task per task and keep it simple: deploy the pilot with the process
- the pilot can be changed without deploying a new process, when a new form is uploaded

Ideas are:

- set the pilot in the "description" because we don't have any way to create some meta data on tasks / process. But it's not possible to update the information without deploy a new process
    - Easy, and no duplication in variable
    - No way to update the information on the server
- set the pilot in a form. Then, each form has is embedded access
    - How to do? No way to set any description / Comment in the form.
    - Not fair for the design: this is the goal of the process designer to manage this information, like the contract
    - The information must stay on the server, and never accessible/ change on the customer side (then we lost the security), so it's not possible to set it in an Asset.
    - Advantages: When we upload a new form, the new pilot are loaded
- Set in a properties file on the BonitaHome
    - How to do? A properties file is saved on the server, and the RestApi Context access it.
    - Advantage: we can set / change at any time the content, via an administrative tool
    - Concern: Deployment is very complex: the administrator should update this information when he deploys a process?
    - Workaround: at the first call, the RestApiContext can update the information in this local database. Or this information hide the first mechanism (if a pilot is describing here, use it, else search the pilot in the process)

13.20.   Can we extends the BDM Search ?

The Rest API getContext will not change, but a new RESTAPI customPage can be used. The idea is to provide the same way for the custom page than in the Form, in order to simplify the usage of the BDM and to protect it.

The RestAPI will be then something like

```
Extention/getPageContext?page=myCustomPage&pilot=searchCity&country=USA&p=0&c=100
```
Give the custom page

The custom page help to find the description of pilot.

Give the pilot

Then in the custom page myCustompage, the pilot to find is searchCity (in order to have multiple pilot : you like to search a city, or maybe search a customer ? So you want to keep this two different call

Give parameters: then, specially if you want to run a request, you need parameters. Give the parameters, and reference them in the pilot

```
{
  "pilotname" [
    "searchCity" : {
      "City":"city?Search=byname&secretcity=False&country={{country}}&p={{p}}&c={{c}}",
      "cityresult" : {
        "name": "data",
        "zipcode": "data",
        "firemenlocation" : { "street" : "data" }
      }
    }
```
Note : to be specify : how describe the request AND the result ?

Aggregate data: on one Rest API, you may want to fetch multiple value from multiple data. Using the Pilot, you can do that.

HEADQUARTERS

PARIS, FRANCE

76 boulevard de la République
92100 Boulogne-Billancourt

EMEA, ASIA & LATIN AMERICA

GRENOBLE, FRANCE

32, rue Gustave Eiffel
38000 Grenoble

NORTH AMERICA

SAN FRANCISCO, USA

44 Tehama Street
San Francisco, CA 94105

NEW YORK, USA

33 Nassau Avenue
Brooklyn NY 11222

Bonitasoft

www.bonitasoft.com