Fix Me! This page is not fully translated, yet. Please help completing the translation.
(remove this paragraph once the translation is finished)

General

Which user may see specific parts (so called sections) within the smartform is determined by the Section Handling. These restrictions can either apply in a whole process or at a certain level in the course of the process. The following permissions can be granted or revoked:

- write permission
- read permission

Preparation of the smartform

The part which should be affected by the Section Handling will be enclosed for example with **DIV** in the smartform, and therefore marks a completed part.
The **DIV** should be given necessarily the attribute class="section". The ID of **DIV** can be chosen optional.
Two DIV-areas are seen in the following example:

- DIV with the ID="section1", contains:
  - Label "Section1"
  - Textarea
- DIV with the ID="section2", contains:
  - Label "Section2"
  - Textarea

Whereat the content within a **DIV** is irrelevant and may be optional.

```
<div class="section" id="section1">
  <label for="section1Input">Section1</label>
  <input id="section1Input" name="section1Input" type="text"></input>
</div>
<div class="section" id="section2">
  <label for="section2Input">Section2</label>
  <input id="section2Input" name="section2Input" type="text"></input>
</div>
```

> If the Section is inside a table (<table>), a tBody should be used instead of a DIV, for defining the Section.

Section Handling Parameter

The parameters for the Section Handling, with which the rights are assigned, are defined in Signavio for the process :

Section Handling Parameter general

```
<section-node-mapping>
  <node-mapping name="Name_of_a_node_from_the_process">
    <section name="ID_of_the_Section_from_the_smartform">
      <read assignment="*"/>
      <write assignment="*"/>
    </section>
  </node-mapping>
</section-node-mapping>
```

- **<section-node-mapping>** : the section-node-mapping element is the major component of the Section Handlings, where the rules of the restriction are
- **<node-mapping>** : the attribute name can be given optionally here. This makes it possible to configure the restrictions that they apply only at a certain level within the process, e.g. User A is allowed to see the textarea, if the smartform is opend during the first node but not once the process is on node B.
  If the attribute name is not given at this point, the restrictions extend to the entire course of the process. In addition, multiple **<node-mapping>** can be specified within the **<section-node-mapping>** element.
- **<section>** to this element must be given the attribute name, which indicates the ID of the desired Section of the smartform. Within the Section element, now the rights can be distributed that will apply for the specified Section.
- **<read>** it can be defined now via the read element, who can see the content within a section. The people are determined via the assignment attribute.
- **<write>** it can be defined now via the write element, who can work on the content within a section (e.g. write in a textarea). The people are determined via the assignment attribute.

Assignment

Following options are now available for the assignments within the read and write elements:

Target group	Notation
individual user	“user(username)”
Swimlanes	“swimlane(Swimlanename)”
Group	“group(Gruppenname)”
Wildcard (for no user)	“”
Wildcard (no restriction)	“*”
Negation(this user can not…)	“! user(…)“

Multiple selections can be realized by semicolons, e.g.:
“group(Sales);group(Controlling);user(Max.Mustermann)”

Locking

Locking can also be specified to the assignments and is used as a lock for editing. For implementation the locking the parameter lockable=“true” must be declared. Locking can be place on the node-mapping or the section itself. The locking attribute of the section always outweighs the attribute of node-mappings.

```
<section-node-mapping>
  <node-mapping lockable="true">
    <section name="section1">
      <read assignment="*"/>
      <write assignment="*"/>
    </section>
    <section name="section2" lockable="false">
      <read assignment="*"/>
      <write assignment="*"/>
    </section>
  </node-mapping>
</section-node-mapping>
```

If now the smartform is opend, the area is locked by default, but behind the locked element is a pen symbol:

Via a click on the symbol, the locked array can be unlocked and therefore edited. The array is locked for all the other users after clicking, which means if an other user opens the smartform it is not possible to unlock and edit the locked array while the array is unlocked by the first user. This is for preventing that input data are accidentally overwritten by an other user. The field can only be unlocked by other users, if the user, who owns the current edit permission, saves and closes the smartform.

Remove pen symbols

If lockable is set on false, a pen sybol appears infront of a section as long as the process is not at the node which was specified in the associated NodeMapping. If the process is then at the stage which is

specified, the pen symbol will be removed. If now all pen symbols shall be removed, a global NodeMapping must be specified, in which all sections are defined once and get an assignment. If there is no change onf the functioning of the smartform, it is enough to give a wildcard to every section in the global NodeMapping.

NodeMapping before instance start

If parts should be limited before instance start, a NodeMapping must be inserted which has the name of the start. If no name is given for the start, the start gets the name **“StartEvent_1”** by default and the end the name **“EndEvent_1”**, which can be used in NodeMapping

---

Section Handling Parameter examples using the smartform examples

The following image shows the initial state of the smartform without Section Handling (everything is visible and editable):

The upper part is in the first section and the lower part in the second section. In the examples all are designed for one user as limited, but it is also possible to carry out all following examples with the assignments from above. The following szenarios can now be generated by using of the Section Handling:

Section visible and editable only for certain users

Now the upper part shall be visible and editable only for a specific user. The used Section Handling parameter for this, might look like this:

```
<section-node-mapping>
  <node-mapping>
    <section name="section1">
      <read assignment="user(TIM)"/>
      <write assignment="user(TIM)"/>
    </section>
    <section name="section2">
      <read assignment="*"/>
      <write assignment="*"/>
    </section>
  </node-mapping>
</section-node-mapping>
```

The smartform would be look like the initial state for the user TIM like for all the others but the following picture would result:

As you can see, only the lower part of the smartform is visible.

Section editable only for certain user

Now the upper part shall be visible, but editable only for the user TIM. The used Section Handling parameter for this, might look like this:

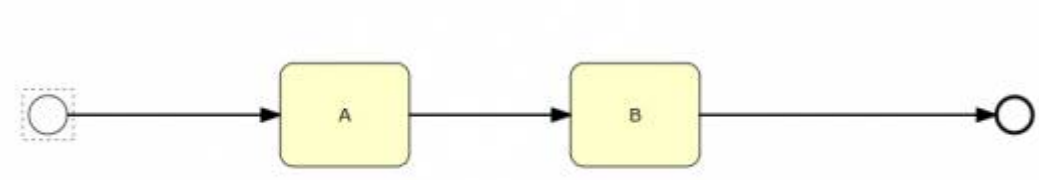
```
<section-node-mapping>
  <node-mapping>
    <section name="section1">
      <read assignment="*"/>
      <write assignment="user(TIM)"/>
    </section>
    <section name="section2">
      <read assignment="*"/>
      <write assignment="*"/>
    </section>
  </node-mapping>
</section-node-mapping>
```

In turn the following image for all other users results, who want to open the smartform:

The upper Section is now visible for everybody, but nothing can be entered in the textarea, because the read parameter is not set.

Section restriction on limited area within the process

The restrictions in the previous examples were global and apply to the entire process. Now it is possible to assert the restriction at certain points in the process. The first example, where only the second section is visible, shall be extended now, that the restriction only applies if the progress of the process is at a particular node. This will be illustrated by the following process:



The desired goal is, that as long as the process is at node A only the user TIM is allowed to see the upper part of the smartform. Once the process is at node B everybody is allowed to see the upper part but not to edit. Therefore for every node section Parameter must be specified. The parameters for achieving the goal are the following:

```
<section-node-mapping>
  <node-mapping name="A">
    <section name="section1">
      <read assignment="user(TIM)"/>
      <write assignment="user(TIM)"/>
    </section>
    <section name="section2">
      <read assignment="*"/>
      <write assignment="*"/>
    </section>
  </node-mapping>
    <node-mapping name="B">
    <section name="section1">
      <read assignment="*"/>
      <write assignment="user(TIM)"/>
    </section>
    <section name="section2">
      <read assignment="*"/>
      <write assignment="*"/>
    </section>
  </node-mapping>
</section-node-mapping>
```

During the whole process, the user TIM can view and edit the entire smartform because there are no restricitions for the user. If the process is on node A, the following image emerges for everybody else when the smartform is opened: Only the lower part is visible.

If the task is now done and the process proceeds to node B, the user TIM can still see everything. For everybody else the smartform looks like this: The upper part is not editable but visible.

How does the Sectionhandling behaves in the smartform search?

In the smartform search only the read-expressions are used which are valid for the whole process.

That means:

If I can see the Section in general, I may even search for it and vice versa.