

---

# Example Dynamic role PEP

## Dynamic role Enforcement Point

### Use case example

We want to define a policy to restrict access to the Soffid console user's page (MainMenu > Administration > Resources > Users).

The users who are assigned to the SOFFID\_RRHH role (from this point forward: *end-users*) will have limitations to perform some actions on the Soffid console users page:

1. The *end-users* only be able to query the information about the users who belong to the same primary group that them.
2. The *end-users* only be able to update the users with internal user type.
3. The *end-users* could not delete any user record.

## XACML Editor

### Policy set

First of all, we define a policy set. We need to define the subject, in that case users with SOFFID\_RRHH role assigned.

## Policy set

Identifier: : TestDynamicPEP

Version: : 1

Description: : TestDynamicPEP

Policy Combining Algorithm: : Permit overrides

## Target

<input type="checkbox"/>	Subjects	Operator	Value	+	<input type="checkbox"/>	Resources	Operator	Value	+
<input type="checkbox"/>	urn:oasis:names:tc:xacml:2.0:subject:role	=	SOFFID_RRHH@soffid						

Displayed rows: 1

<input type="checkbox"/>	Actions	Operator	Value	+	<input type="checkbox"/>	Environments	Operator	Value	+
--------------------------	---------	----------	-------	---	--------------------------	--------------	----------	-------	---

Displayed rows: 0

## Obligations

<input type="checkbox"/>	Obligation	Full fill on	Attribute	Value	+
--------------------------	------------	--------------	-----------	-------	---

Displayed rows: 0

Then, we can create the policies, in that case, we can create three policies, one for each operation that we want to manage.

⊖	TestDynamicPEP (1)	TestDynamicPEP
⊕	UsersDelete (1)	UsersDelete
⊕	UsersQuery (1)	UsersQuery
⊕	UsersUpdate (1)	UsersUpdate

# Policies

We can define a policy for each operation, to permit or deny access.

Also, we can define a variable that contains the *end-user* primary group in order to use it when we define the conditions.

## Policy1

“ The *end-users* only be able to query the information about the users who belong to the same primary group that them.

We need to define two rules, one to permit and other to deny access.

## Policy

Identifier :

Version :

Description :

Rule Combining Algorithm :

## Target

<input type="checkbox"/>	Subjects	Operator	Value	+	<input type="checkbox"/>	Resources	Operator	Value	+
Displayed rows: 0					Displayed rows: 0				
<input type="checkbox"/>	Actions	Operator	Value	+	<input type="checkbox"/>	Environments	Operator	Value	+
Displayed rows: 0					Displayed rows: 0				

## Variables

<input type="checkbox"/>	Variable	Expression	+
<input type="checkbox"/>	group	primary group	
Displayed rows: 1			

## Rules

<input type="checkbox"/>	Rule	Description	Effect	+
<input type="checkbox"/>	PermitQuery	PermitQuery	Permit	
<input type="checkbox"/>	DenyQuery	DenyQuery	Deny	
Displayed rows: 2				

## Obligations

<input type="checkbox"/>	Obligation	Full fill on	Attribute	Value	+
Displayed rows: 0					

# Rules

We define the rule that allow to the *end-user* to query users information who belong to the same primary group that the end-user.

**Rule**

Rule :

Description :

Effect :

**Target**

<input type="checkbox"/>	Subjects	Operator	Value	+	<input type="checkbox"/>	Resources	Operator	Value	+
Displayed rows: 0					Displayed rows: 1				
<input type="checkbox"/>	Actions	Operator	Value	+	<input type="checkbox"/>	Environments	Operator	Value	+
Displayed rows: 1					Displayed rows: 0				

<input type="checkbox"/>	urn:com:soffid:xacml:action:method	=	query	
--------------------------	------------------------------------	---	-------	--

**Conditions**

<input type="checkbox"/>	Condition	Expression	+
<input type="checkbox"/>	Condition	( One and only(group) == One and only(null /primaryGroup/name) )	
Displayed rows: 1			

Then, we define the rule that denies access to *end-users* to query users information.

**Rule**

Rule : DenyQuery

Description : DenyQuery

Effect : Deny

**Target**

Subjects	Operator	Value
Displayed rows: 0		

Resources	Operator	Value
com:soffid:iam:xacml:1.0:resource:s offid-object	=	user
Displayed rows: 1		

Actions	Operator	Value
urn:com:soffid:xacml:action:method	=	query
Displayed rows: 1		

Environments	Operator	Value
Displayed rows: 0		

**Conditions**

Condition	Expression
Displayed rows: 0	

## Policy 2

“ The *end-users* only be able to update the users with internal user type.

We need to define two rules, one to permit and other to deny access.

**Policy**

Identifier : UsersUpdate

Version : 1

Description : UsersUpdate

Rule Combining Algorithm : Permit overrides

**Target**

Subjects	Operator	Value
Displayed rows: 0		

Resources	Operator	Value
Displayed rows: 0		

Actions	Operator	Value
Displayed rows: 0		

Environments	Operator	Value
Displayed rows: 0		

**Variables**

Variable	Expression
Displayed rows: 0	

**Rules**

Rule	Description	Effect
PermitUpdate	PermitUpdate	Permit
DenyUpdate	DenyUpdate	Deny
Displayed rows: 2		

**Obligations**

Obligation	Full fill on	Attribute	Value
Displayed rows: 0			

## Rules

We define the rule that allow to the *end-users* to update users information who are internal users.

Rule

Rule :

PermitUpdate

Description :

PermitUpdate

Effect :

Permit

Target

Subjects

Operator

Value

Displayed rows: 0

Resources

Operator

Value

Displayed rows: 2

Actions

Operator

Value

Displayed rows: 1

Environments

Operator

Value

Displayed rows: 0

Conditions

Condition

Expression

Displayed rows: 0

Then, we define the rule that denies access to *end-users* to update users information.

Rule

Rule :

DenyUpdate

Description :

DenyUpdate

Effect :

Deny

Target

Subjects

Operator

Value

Displayed rows: 0

Resources

Operator

Value

Displayed rows: 1

Actions

Operator

Value

Displayed rows: 1

Environments

Operator

Value

Displayed rows: 0

Conditions

Condition

Expression

Displayed rows: 0

## Policy 3

“ The *end-users* could not delete any user record.

We need to define only one rule to deny access.

## Policy

Identifier :

Version :

Description :

Rule Combining Algorithm :

## Target

<input type="checkbox"/>	Subjects	Operator	Value	+	<input type="checkbox"/>	Resources	Operator	Value	+
Displayed rows: 0					Displayed rows: 0				
<input type="checkbox"/>	Actions	Operator	Value	+	<input type="checkbox"/>	Environments	Operator	Value	+
Displayed rows: 0					Displayed rows: 0				

## Variables

<input type="checkbox"/>	Variable	Expression	+
			Displayed rows: 0

## Rules

<input type="checkbox"/>	Rule	Description	Effect	
<input type="checkbox"/>	DenyDelete	DenyDelete	Deny	

Displayed rows: 1

## Obligations

<input type="checkbox"/>	Obligation	Full fill on	Attribute	Value	
					Displayed rows: 0

# Rules

We define the rule that deny to the *end-user* delete any user.

**Rule**

Rule :

Description :

Effect :

### Target

<input type="checkbox"/>	Subjects	Operator	Value	+	<input type="checkbox"/>	Resources	Operator	Value	+
Displayed rows: 0					Displayed rows: 1				
<input type="checkbox"/>	Actions	Operator	Value	+	<input type="checkbox"/>	Environments	Operator	Value	+
Displayed rows: 1					Displayed rows: 0				

<input type="checkbox"/>	urn:com:soffid:xacml:action:method	=	delete	
--------------------------	------------------------------------	---	--------	--

### Conditions

<input type="checkbox"/>	Condition	Expression	+
Displayed rows: 0			

# Download XML

You can download a XML file with the example: [policy-TestDynamicPEP.xml](#)

# Configure PEP

## Dynamic role Policy Enforcement Point

Enable XACML Policy Enforcement Point : ☒ Yes ☐ No

Policy Set Id :

Policy Set Version :

Trace requests : ☐ Yes ☒ No

---

Revision #20

Created 4 August 2021 13:48:52 by pgarcia@soffid.com

Updated 6 August 2021 08:42:09 by pgarcia@soffid.com