

Process Engine Automation -Developer Guide

Owner: Resultmaker Research & Development Version: 6.0.0 Revision: 2012-11-15

John Thaysen (JT), Michael Aagaard Pedersen (MAP), Lars Pedersen (LP)



Contents

1	I	troduction				
2	I	Installation	.4			
3	I	Installation content	. 5			
	3.1	ProcessEngineAutomation.dll	. 5			
	3.2	2 ResultmakerProcessStarter.exe	. 5			
	3.3	8 ReadThis.txt	.7			
	3.4	Resultmaker.ico	.7			
	3.5	ResultmakerProcessStarter.exe.config	.7			
	3.6	ProcessEngineBatch.bat	.7			
4	I	How-to schedule Process Engine tasks	.8			
	4.1	Using the Task Scheduler on Windows 98, NT, 2000, XP, and Server 2003	.8			
	4	4.1.1 Add a task and select the ResultmakerProcessStarter.exe program	.8			
	4.2	Using the Task Scheduler on Windows Vista, Server 2008, and 7.	12			
5	I	How-to Programmatically Automate The Process Engine	16			
	5.1	How to start a workflow	16			
	5.2	2 How to resume a workflow	17			
	5.3	B How to fill a form in an instance of a workflow.	17			
	5.4	How to get a form from an instance of a workflow	19			
	5.5	How to signal an activity to autoclose	19			
	5.6	How to signal a workflow to autoclose all eligible activities	20			
	5.7	7 How to delete a workflow	21			

1 Introduction

This document, describes shortly how to use Process Engine Automation to start a workflow and fill forms on the started workflow using the Windows Task Scheduler, and how one programmatically can use the ProcessEngineAutomation.dll to start, resume and fill forms on a Process Engine.



2 Installation

The Process Engine Automation is installed on the Process Engine Server as a part of the Process Engine installation, and has the following default location: "C:\Program Files\Resultmaker\ProcessEngineAutomation".

When needed the Process Engine Automation can be installed on any .Net PC/Server_by running the installation file: "ResultmakerProcessEngineStarter.msi".

When used programmatically you can just reference the ProcessEngineAutomation.dll.

This documentation covers version Process Engine Automation 1.1.0 or later.



3 Installation content

The following files are installed as part of the Process Engine Automation installation:

3.1 ProcessEngineAutomation.dll

This DLL can be used to programmatically access the Process Engine.

3.2 ResultmakerProcessStarter.exe

An executable used to start a workflow in the Process Engine.

This executable has the following arguments:

ResultmakerProcessEngineStarter.exe /W: [/F:] [/S:] [/U:] [/T:] [/D:]

-REQUIRED-

/Workflow:<name> or /WorkflowName:<name>

The name of the project to be started. Short form is '/w'.

- OPTIONAL -

/FormFilling:<string> or /FormFillingCollection:<string>

The FormFillingCollection is used to fill forms on a workflow. The string must contain a collection of activity references as well as form element names and values to be filled. The format is like this:

/FormFillingCollection:"ActivityReference=Transaction\WorkflowGroup\FormActivity|TextField0=Va lue 1|TextField1=Value

2&ActivityReference=Transaction\WorkflowGroup\FormActivity2|TextField0=Value 3|TextField1=Value 4|"

The FormFillingCollection contains a collection of one or more strings separated by the &-character. Thise strings contain an ActivityReference element with the format:

Transaction\WorkflowGroup\FormActivity and a section with name of the form question elements and the values to be filled out, the format is like this: |TextField0=Value 1|TextField1=Value 2. Short form is '/f'.

/Security:<token> or /SecurityToken:<token>

The SecurityToken is used to login to the Process Engine. If not applied the default value is the windows user credentials, which are delegated to the Process Engine. Short form is '/s'.

/Uri:<uri>

An uri to the Process Engine process.aspx page. A default value is stored in a config file App.config and do not have to be applied. Short form is '/u'.

/Temp:<folderName> or /TempFolder:<folderName>

Folder name of folder in the users temp folder, used to dump input and output files if activated. Default temp folder is stored in the App.config file and do not have to be applied.



An example of the temp folder and file name is like this:

"C:\Users\UserName\\AppData\Local\Temp\[folderName]\[WorkflowName]_d9f33c76-bb6b-4ec3-a161-8f2e363be34d.xml".

Short form is '/t'.

/DumpInputOutput:<boolean> or /Dump:<boolean>

Activates or deactivates the dumping of input and output file in temp store. Default value is located in App.config and do not have to be applied. Short form is '/d'.



3.3 ReadThis.txt

This file contains a description of the required/allowed arguments for the executable "ResultmakerProcessEngineStarter.exe", similar to the one given in section 3.2.

3.4 Resultmaker.ico

The resultmaker icon.

3.5 ResultmakerProcessStarter.exe.config

Configuration file, which makes it possible to setup default values for most of the parameters already described. The following is an example showing all the setting parameters.

```
<setting name="ProcessEngineServer" serializeAs="String">
        <value>127.0.0.1</value>
</setting>
<setting name="ProcessEngineServerTransport" serializeAs="String">
       <value>http</value>
</setting>
<setting name="ProcessEngineServerPort" serializeAs="String">
       <value>80</value>
</setting>
<setting name="ProcessEngineServerResource" serializeAs="String">
       <value>/OC/5.2/process.aspx</value>
</setting>
<setting name="DefaultSecurityToken" serializeAs="String">
       <value>WinDelegation:OUTOFBANDPAYLOAD==</value>
</setting>
<setting name="DumpInputOutput" serializeAs="String">
       <value>False</value>
</setting>
<setting name="TempDumpFolder" serializeAs="String">
             <value>Resultmaker/ProcessEngineAutomation</value>
</setting>
```

Notice, the special value of the DefaultSecurityToken = "WinDelagation:OUTOFBANDPAYLOAD==", which is used to indicate that as default we use Windows Credential Delegation to delegate ones windows credentials on executing command to the Process Engine.

3.6 ProcessEngineBatch.bat

Small batch file example, could be used if one prefer to execute the ResultmakerProcessStarter.exe including arguments from a batch file.



4 How-to schedule Process Engine tasks

One can use the Windows Task Scheduler to schedule Process Engine tasks, in the following two different versions of the Windows Task Scheduler are explained.

4.1 Using the Task Scheduler on Windows 98, NT, 2000, XP, and Server 2003.

To use the scheduling service on Windows 98, NT, 2000, click on the My Computer icon located on the Desktop. Then double-click on Control Panel to get to the Scheduled Tasks folder.

On Windows XP and Server 2003 you can access this from the Start Menu and clicking on Settings and then Control Panel to Scheduled Tasks.

Double-click Add Scheduled Task. Follow the instructions in the Add Scheduled Task wizard.

4.1.1 Add a task and select the ResultmakerProcessStarter.exe program Click Add Scheduled Task.

Scheduled Task Wizard	×
	This wizard helps you schedule a task for Windows to perform. You select the program you want Windows to run, and then schedule it for a convenient time. Click Next to continue.
	< <u>B</u> ack <u>Next></u> Cancel

Click Next



	Application	Version	+
	🖊 🛛 🥂 Accessibility Wizard	5.2.3790.395	
	🔄 🔍 Address Book	6.00.3790.39	
	📷 🛛 🛃 Backup	5.2.3790.395	
	BM Broker Monitor		
	BrokerTester		
- The	BT BrokerTools		
		F 2 2790 0 (~	-
10 V 1		Browse	

Click Browse

Select Program	to Schedule				? ×
Look jn:	ProcessEng	ineAutomation	•	3 🕫 🖻	
My Recent Documents Desktop My Documents My Computer	ProcessEngini ProcessEngini ReadThis.txt Resultmaker.i	eAutomation.dll eBatch.bat ico rocessStarter.exe rocessStarter.exe.confi	g		
My Network Places	File <u>n</u> ame: Files of <u>type</u> :	ResultmakerProcess Programs	Starter.exe	<u>*</u>	<u>O</u> pen Cancel

Select

c:\Program Files\Resultmaker\ProcessEngineAutomation\ResultmakerProcessStarter.exe

Click Open

	<u>I</u> ype a name for this task. The task name can be the same name as the program name.
1	Transfer money task
4	Perform this task:
	C Daily
1	
	C Monthly
-	C One time only
1	C When my computer <u>s</u> tarts
	C When I log on
	Z Back Nevt Cancel

Type in name that describes the task, and choose how often the task has to be done.

In this example we choose weekly, click Next.

	Start time:	a day you want this task to start.
·	JEE:47	-
n	Every 1 🛨	weeks
	Select the day(s) o	of the week below:
~ ~ ~	🔲 <u>M</u> onday	Thursday
	🗖 Tuesday	🔽 <u>F</u> riday
	🔲 <u>W</u> ednesday	🔲 <u>S</u> aturday
NP CON		🔲 S <u>u</u> nday

Select additional timing options, here the options for a weekly task is shown, click Next.



	Enter the name and pa run as if it were started	issword of a user. The task will by that user.
· 0	Enter the user name:	ACROPOLIS\it
1	Enter the password:	•••••
	<u>C</u> onfirm password:	•••••
	If a password is not en not run.	tered, scheduled tasks might
	< <u>B</u> ack	Next > Cancel

The execution of the task in the task scheduler needs to be executed as a windows user, add here the user name and password.

Use a system/machine user with a never expiring password; else the task will stop working, when the password expires.

Scheduled Task Wizard		×
	You have successfully scheduled the following task: Transfer money task	
- 9	Windows will perform this task: At 09:47 every Fri of every week, starting 08-09-2009	
	☑ Open advanced properties for this task when I click Finish.	
	Click Finish to add this task to your Windows schedule.	
	< <u>B</u> ack Finish Cancel	

Set the check mark "Open advanced properties to add arguments to the execution". Click Finish.



Run:	esultmakerProcessStarter.	.exe" /w:AUTOMATION1 /\$1
_		<u>B</u> rowse
S <u>t</u> art in:	"C:\Program Files\Resultr	maker\ProcessEngineAutomatio
<u>C</u> omments:		
		<u>S</u> et password
Bun as:		Der passinora

Add arguments to the program in the text field named "Run".

Important: The command line options must be placed OUTSIDE of the "" that surround the path. Uncheck the "Run only if logged on" else it won't run if the user is not logged on.

Click Apply and Click OK (you probably have to type in password again if you changed anything).

Tip: Alternatively you can point to a Batch File that contains the specific command line instructions to start the software. Also, with batch files you can conveniently execute several commands in a sequence.

4.2 Using the Task Scheduler on Windows Vista, Server 2008, and 7.

In this example all the possible options are not shown, only a simple example.

Start The Task Scheduler (For instance type "Task Scheduler" in the start search at the start menu).



Action view Help				
🔶 🔲 🔽 🖬				
Task Scheduler (Local)	Task Scheduler Summary (Last refreshed: 08-09)-2009 10:13:23)		Actions
	Overview of Task Scheduler Wou can use Task Scheduler to creat will carry out automatically at the taction menu. Tasks are stored in folders in the Ta operation on an individual task, sel on a command in the Artion menu. Task Status Status of tasks that have started in the follow Summary: 55 total - 0 running, 55 succeed	te and manage commo mes you specify. To be sk Scheduler Library. To ext the task in the Task wing time period: ed, 0 stopped, 0 failed	n tasks that your computer gin, click a command in the view or perform an Scheduler Library and click Last 24 hours	Connect to Another Computer Create Basic Task Create Task Create Task Import Task Display All Running Tasks AT Service Account Configurati View Refresh Help Help
	Task Name Rur ⊞ Consolidator (last run succeade ⊞ CrawlStartPages (last run succe ⊞ GoogleUpdateTaskUser5-1-5-2 ⊞ GoogleUpdateTaskUser5-1-5-2 ⊞ SR (last run succeeded at 08-09 Ш Sr.utaseTash (last run succeaded at 08-09)	n Result Run Start	Run End	
	Active Tasks Active tasks are tasks that are currently en Summary: 20 total	abled and have not exp	ired.	
		1 D T'	Trigger	
	Task Name Ne	xt Kun Time	inggers	

Click Create Task in the Actions pane.

Create Task					
General Triggers	Actions Conditions Settings				
Na <u>m</u> e:	Daily web page updating delivery				
Author:	ACROPOLIS\jt				
<u>D</u> escription:	This is and example, in the description field you should	well describe the task.			
Security options					
When running the task, use the following user account:					
ACROPOLIS\jt Change User or Or Run only when user is logged on		Change <u>U</u> ser or Group			
 Run whether user is logged on or not 					
🔲 Do not st	ore <u>p</u> assword. The task will only have access to local con	nputer resources.			
Run w <u>i</u> th highest privileges					
✓ Hidd <u>e</u> n	<u>C</u> onfigure for: Windows Vista [™] or Windows Server	™ 2008 👻			
		OK Cancel			

The recommended settings are shown, Use a system/machine user with a never expiring password; else the task will stop working, when the password expires.

Select the tab Triggers, and click New



県 New Trigger	
Begin the task: Settings	On a schedule 🔻
One time	Start: 08-09-2009 To:26:09 Synchronize across time zones
© <u>D</u> aily	
© <u>W</u> eekly	
© <u>M</u> onthly	
Advanced settings	
Delay tas <u>k</u> for u	up to (random delay): 1 hour 👻
🔲 Re <u>p</u> eat task ev	ery: 1 hour
Stop all ru	nning tasks at end of repetition durat <u>i</u> on
🔲 Stop task if it r	uns longer than: 3 days
Expire:	08-09-2010 □▼ 10:26:10 Synchroniz <u>e</u> across time zones
<table-cell> Ena<u>b</u>led</table-cell>	
	OK Cancel

There are many option in the dropdown "Begin the task"; here we only show "on a schedule". After you have setup the trigger, click "OK".

neral Triggers	Actions Conditions Settings	
When you create	a task, you can specify the conditions that will trigge	er the task.
Trigger	Details	Status
One time	At 10:26 on 08-09-2009	Enabled
<u>N</u> ew	<u>E</u> dit <u>D</u> elete	

Select the tab Actions, and click New.



New Action		
You must spe	cify what action this	task will perform.
Action:	Start a program	•
Settings		
Program/so	:ript:	
rocessEngin	eAutomation\Result	makerProcessStarter.exe" B <u>r</u> owse
<u>A</u> dd argum	ents (optional):	/w:AUTOMATION1 /s:1
S <u>t</u> art in (opt	tional):	
		OK Cancel

Select Start a program, and add the Program "C:\Program

Files\Resultmaker\ProcessEngineAutomation\ResultmakerProcessStarter.exe", and add at least the /workflow: argument to start a workflow.

Click OK.

The tabs conditions and settings contains additional options which we do not need to change is this example.

Click "OK" and type in the user password.



5 How-to Programmatically Automate The Process Engine

The assembly ProcessEngineAutomation.dll can be used to start, resume and fill forms in the Process Engine.

This assembly contains two public classes and three custom exception classes, the main class is named ProcessEngineFacade and represents the available Process Engine functionalities, and the other is named ProcessEngineForm and represents a Form in the Process Engine. Se the class diagram shown below:

Fields	ProcessEngineForm						
🧬 commandHandler	→ Dictionary <string string=""></string>						
Properties	stang, sang						
DumpInputOutput	Properties						
ProcessEngineUri	MoveNext						
TempFolder	PageNo						
Methods	□ Methods						
CreateProcessEngineForm	🕸 Clone						
■ FillForm (+ 1 overload)	🔊 ICloneable.Clone						
ProcessEngineFacade (+ 2 overloads)	ProcessEngineForm (+ 2 overloads)						
ResumeWorkflow (+ 1 overload)	a [™] ToInterviewAnswers						
StartWorkflow (+ 1 overload)							
ProcessEngineException S	mmunicationException 🛞 ResponseParseE	ception					
Class Cla	ess Class						
→ Exception → I	Exception → Exception						

In the following, an example is given of how to start, resume and fill a form by using these classes.

5.1 How to start a workflow

The following code example shows how a workflow can be started:

```
var facade = new ProcessEngineFacade(new
Uri("http://octest02/0C4/process.aspx"));
```

int workflowId = facade.StartWorkflow("WorkflowExample");

The first code line creates a new ProcessEngineFacade, this constructor demands at least one parameter an URI containing an URL to the Process Engine process.aspx page.

In this example we are using windows credential delegation to delegate our windows credentials to the Process Engine, if we need to use another login strategy one has to use the overloaded constructor including a parameter called security token.

For debugging purposes there are also two properties where one can enable the InputOutputDump and setup the temp folder wherein all input and output commands to the Process Engine can be dumped as XML.

In the second line the workflow with the name "WorkflowExample" is started.



The "StartWorkflow" method returns an integer which is the id of the just started workflow instance; this id is used as a handle to the just started instance of the workflow.

5.2 How to resume a workflow

In this example we expect that an instance of a workflow is already started somehow, and that we somehow have got the workflow id:

```
var facade = new ProcessEngineFacade(new
Uri("http://octest02/0C4/process.aspx"), "1")
{
    DumpInputOutput = true,
```

TempFolder = @"Resultmaker\ProcessEngineAutomation"

};

```
facade.ResumeWorkflow(workflowId);
```

First we have to create a ProcessEngineFacade to be able to communicate with the Process Engine, in this example we have added a second parameter a security token "1", which is the "demo" user on the present Process Engine 5.2 system (the security system will be changed in the new 6.0 release, and it will therefore not necessarily be possible just to specify the UMSId in later versions of the Process Engine).

Moreover, we have enabled dump of commands, which dumps the commands send as XML to the specified folder in the user's temp folder.

Next, we call the "ResumeWorkflow" method with the workflow Id as the first parameter, and if no exceptions are thrown the resuming of the workflow went okay without error.

5.3 How to fill a form in an instance of a workflow.

In the following example we start a workflow and fill the first page of a specified activity. Moreover, we use delegation of windows credentials, which is used if no security token is specified.

```
var facade = new ProcessEngineFacade(new
Uri("http://octest02/0C4/process.aspx"));
int workflowId = facade.StartWorkflow("WorkflowExample");
var form = new ProcessEngineForm();
form.Add("LastName", "Ackerman");
form.Add("FirstName", "Peter");
var activityId = new[] { "Transaction", "WorkflowGroup",
"FormActivity" };
facade.FillForm(workflowId, activityId, form);
Below this WorkflowExample is shown in the Process designer.
```

The strings for the "activityId" string array are taken from the "Internal ID" properties of the Transaction, Group and activity level in the Workflow Designer. In the figure below the properties for the transaction level are shown.



🚰 Properties	×		🗞 NewWorkflow *	
● ≣ 2 ↓ □	-	·	WorfklowExample	Role1 🕹
🗆 Basic Data			11 New Group	
Description				
Internal ID	Transaction		1.1.1 New Form Activity	2
Text	New Transaction			
Extensibility			'	
Custom Data				
Scripts				
🗆 Form data	=			
AutoGenerateOioXml	True			
GenerateSingleSchema	False			
OioXmlExportName	OC_Q_400002_Xslt			
Misc				
FormId	400002			
Namespace	tmp			
NameSpaceUri	http://rep.oio.dk/virk.dk/tmp/sc			
Predecessors				
PurposeRule				
Receipt Email				
ReceiptMailBody	The data submitted is attac			
ReceintMailFormat	Text			
Internal ID				
The internal ID of the Workfl	ow Item.		Add Transaction	

First a new Process Engine façade is created, and a new workflow instance of the workflow with the name "WorkflowExample" is created. An workflow id is received as a return value.

Thereafter a ProcessEngineForm is created, the default constructor is the first page form where the pageNo = 0 and MoveNext = 1, if one uses more than one <u>form page per activity</u> one must use the overloaded constructor to specify the pageNo and MoveNext parameters of the form.

We then add two fields to the form (field names are "LastName" and "FirstName"). These are taken from the form element's "Internal ID" properties.

A string array named ActivityId is created and instantiated as an array of strings containing names identifying the activity to fill out.

Below the form used in this example is shown in the Process Designer:



🛃 Auto1 - octest02 *		×		Layo	ut													×
E Automation 1			TestAutomation\First Page															
E First Page 						_												
				Test	af F	illing	For	m										
		=		First	nan	ne												
			ģ) First N	lame				¢.									
ab Field 4			Last name								•							
				LastN	lame				· ·	•		•	•	•			•	
					•	• •			•	•	•	•	•	•	• •	•	•	
				Field	3				• •	•	•	·	•	·	• •	•	·	
		-		TextF	ield()			· ·	•	•	•	•	•		•	•	
				Field	4					•	•	•	•	·		•	·	-
Properties		23		TextF	ield1				· ·	•	·	·	•	·		•	·	=
		•		· ·	·	• •	•	·		•	•	•	•	•			•	
₽ 2 □					•	• •	•	•			•		•			•	•	
🗆 Basic Data					•	• •	•	•	• •	•	•	·	•	•	• •	•	•	
Description					•	• •	•	•	• •	•	•	•	•	•		•	·	
Internal ID	First Name	_			•	• •	•	•	• •	•	•	•	•	•		•	•	
Show Icons	Irue Cent como	- 11			•	• •	•	•	• •	•	•	•	•	·		•	·	
Type	Text				·	• •		·		•	·	·	·	·		·	·	
Display Format	1 2012	Ξ				· ·	•	•						·			•	
No. of Decimal Places	-1					· ·												
Show Thousand Separator	False					· ·												
UnChange	1																	

Last the FillForm method is called, with no security token specified, hence delegation of windows user credentials is used.

If no exception is raised the filling of the form went fine.

5.4 How to get a form from an instance of a workflow

In this example we expect that an instance of a workflow is already started somehow, and that we have got the workflow id:

```
var activityId = new[] { "Transaction", "WorkflowGroup",
"FormActivity" };
```

var form = facade.GetForm(workflowId, activityIdentification);

A string array named ActivityId is created and instantiated as an array of strings containing names identifying the activity to get the first form of the specified activity.

Thereafter we call GetForm and a form with the present values of this instances form is returned.

The returned form object can be enumerated, and the form field names and the values in the fields can be retrieved.

```
foreach (var keyvalue in form)
{
    string formFieldName = keyvalue.Key;
    string formFieldValue = keyvalue.Value;
}
```

5.5 How to signal an activity to autoclose

A workflow can be signaled to autoclose an activity. Autoclosing means that the activity completes as if the form attached to it (if any) were filled out with the values already available from prefilling with workflow and user variable mappings. An



activity cannot autoclose unless the "AttemptAutoClose" property on the activity is set to true, or if any of the questions set to "required" on the form cannot be prefilled.

🍣 NewWorkflow *			Properties		×
New Workflow	Role1 🕹				•
1 New Transaction	(Add)		Z + ==		
1 1 New Group	(Add)		Basic Data		
			Description	-	- 11
New Form Activity	4		Internal ID	FormActivity	-
			lext	New Form Activity	
			Extensibility		4.5
		L 🕂	Custom Data		-
			Scripts		
			Form data	-	
			AutobienerateFormXmlSchema	I rue	- =
			FormXmlExportName	DefaultQuestionnaire_Schema	- 11
			Generate Receipt	True	- 11
			ReceiptExportName	DefaultQuestionnaire_XsltReceipt	
			Misc		
			AttemptAutoClose	True 💽	
			EnableInterviewSession	False	
			Form ID		
			Predecessors		
			PurposeRule		Ŧ
		At	ttemptAutoClose		
			•		
Add Transaction					

In this example we suppose that a workflow has already been started, that we have the workflow id, and that we have a ProcessEngineFacade object properly initialized to operate on the process engine that the workflow resides on:

```
int activityId = facade.GetActivityId(workflowId, "Transaction",
"WorkflowGroup", "FormActivity");
```

bool didAutoClose = facade.AutoCloseActivity(workflowId, activityId);

As in the previous example we get the activityId by navigating through the activities comprising the path to the desired activity by their InternalID property.

We then call the facade's AutoCloseActvity metod with the workflow id and the activity id as parameters.

The method returns a bool which indicates whether or not the activity was in the closed state after the operation. Note that this is the case even if the operation had no effect because the activity was already closed.

Attempting to autoclose an activity to which the security token does not provide write access results in an error.

5.6 How to signal a workflow to autoclose all eligible activities

A workflow can be signaled to close all activities eligible for autoclosing. An activity is eligible for autoclosing if it's "AttemptAutoClose" property is set to true and if it is possible to prefill all required questions on the attached form of the activity (if any). In addition, the executing identity as conveyed by the security token passed to the process engine must have write access to an activity in order for it to be eligible.

In this example we assume that a workflow has already been started, that we have the workflow id, and that we have a ProcessEngineFacade object properly initialized to operate on the process engine that the workflow resides on:

var activityIds = facade.AutoCloseAll(workflowId);

We call the facade's "AutoCloseAll" method, supplying the workflow id. The method returns an IEnumerable<int> with the activity ids of any activities that were



closed as a result of the autoclose attempt. Only the ids of those activities that were not in the closed state before the command but are in that state after its execution are returned.

5.7 How to delete a workflow

In the following example we delete a workflow based on the worklowld.

```
var facade = new ProcessEngineFacade(new
Uri("http://octest02/OC4/process.aspx"));
facade.DeleteWorkflow("WorkflowExample");
```

