ActiveFlow Designer Guide

Release Date: July 12, 2009

Contents

Introduction to ActiveFlow

The **KAISHA-Tec BPM system** has two main components. ActiveModeler Avantage, the process definition and analysis tool and ActiveFlow, the workflow engine.

In this introduction, we show how ActiveFlow can help you reduce costs by automating manual processes and at the same time maintaining a strong audit trail of "who did what and when".

What exactly do you we mean by process definition and workflow?

Process Definition: This is the representation of the activities that make up a business process, together with the organization structure, role interactions, metrics, and information flow involved in the process.

The workflow definition is an extension of the basic process definition and Avantage can be used for this as well.

Workflow: This is the automation of a process, in whole or part, during which documents, information or jobs are passed from one participant to another for action, according to a set of procedural rules as defined in the process definition.

Workflow enables you to automate administrative processes in diverse industries ranging from commerce and government to manufacturing and health care.

Similarly, the range of possible applications includes purchase requisitions, order forms, personnel forms, expense reports, time cards, sales forms, problem reports, financial consolidation, and manufacturing ticket followers.

The range of both the industries and the applications themselves is limited only by your imagination. Remember also that the information flow can be both within a company (intranet based), a global Internet workflow, or a combination of both and can involve customers, suppliers as well as internal staff.

Unique Workflow design studio

ActiveModeler Avantage and ActiveFlow are tightly integrated. Avantage is based on our successful first-generation product called KAISHA Modeler Pro, which sold tens of thousands of copies. There was a substantial investment in Quality Assurance with this product both from KAISHA-Tec and our Japanese partners, NEC. Workflow automation is about automating an existing or defined process, whereas process modeling is a means of depicting and measuring a process; leading to the next step of process improvement. Adding ActiveFlow to this powerful process modeling tool gives you the power to create a documented, correct, and automated business process. ActiveModeler Avantage is a sophisticated yet easy-to-use software package following the latest BPMN (Business Process Modeling Notation)standard from the OMG group. It represents products, departments, and roles clearly and simply, with drill-down to hide complexity.

You can generate a workflow very quickly using the Avantage Workflow Wizard, with just a few mouse clicks. The workflows are standardized, so you avoid expensive and error-prone custom workflow programming.

ActiveFlow was first developed for the Japanese market, so despite being easy to use, it can meet the most complex and demanding jobs. Japan is sometimes called the "Home of Workflow", with each action being authorized by "the group". Many kinds of workflow have been developed in Japan and consequently we feel ActiveFlow can easily cope with the workflows in Western companies rather easily, which tend to be a subset of the Japanese requirements.

Benefits of workflow

Workflow has major benefits for an organization. In particular:

- Workflow formalizes a business process; you can be confident all steps have been followed correctly with validity and audit checks as defined by the workflow designer. Staff cannot miss out steps. Irrelevant work is also controlled and time is not wasted on non-defined jobs
- With automation and reproducibility, you can achieve volume insensitivity. As your business transaction volumes grow, you do not need a linear growth in the number of staff
- Productivity improves as staff can be assigned to more important and meaningful work
- Paperwork and paper-chasing are eliminated
- Improved tracking options: A customer or staff member can instantly know the status of any work item. The "who, when, where questions" are answered by the workflow
- Interfaces to external databases enable validity checks and external process interactions to be automated and streamlined
- Decisions that were made by people can be made by the workflow, based on the same decisions human staff were making
- Business efficiency can be more accurately measured: you can easily see how much work was done each day
- The workflow system links to an organization database so you know who does what.
- Security is assured by predefined user rights, which give access only to those who need it
- A proper audit trail shows each work item, what was done, when, and by whom.

Benefits of ActiveFlow

ActiveFlow offers a very high level of functionality yet is completely in tune with real-world requirements, a few example of which are shown in the **Real-world requirements** table below. This is because we work very closely with our customers.
For a full feature list please go to http://www.activemodeler.com/activeflowFeatureList

ActiveFlow is web-based, so the workflow participants can be anywhere in the world: on the corporate Intranet, dispersed on the Internet, or a combination of the two.

As an example, an electric company might use ActiveFlow to enable customers to apply for a new connection through the Internet. Customers could fill out an application form online over the Internet and optionally include a map (as an attached file). The application is sent automatically to the electric company head office where a clerk checks this information. The supply is approved on the corporate Intranet via a corporate workflow process and result (including the connection time and other details if successful) is sent to the customer via the Internet, with an additional email notification. This whole application could be built within hours with ActiveModeler and ActiveFlow, and ActiveFlow would provide the production workflow control.

ActiveFlow is based entirely on the latest Microsoft technology and uses no proprietary forms or routing engine. The workflow model also conforms to the Workflow Management Coalition (WfMC) standards.

This protects any investment you make in workflow today. We will move with future technology changes and you will not be locked into a proprietary system, as with many other workflow systems (which may be unsuitable for tomorrow's technologies).

We use the following components:

Client

- Client forms can be viewed with Microsoft Internet Explorer, Firefox, Google Chrome or the Macintosh Safari browser. There is **no** software or ActiveX to be installed on the client side.

Server components

- Windows 2003 Server or above.
- Microsoft Internet Information Server (IIS).
- SQL Server 2000 and above for the database engine.
- Access to an SMTP Mail (optional).

Client-side development

- Client forms are developed with any HTML or WebForms (ASP.NET) editor

Real-world workflows

We all know that Workflow is about moving documents and information between roles, and controlling and tracking that movement. However in the real world, simple movement from A to B to C is not sufficient. Many factors may arise to increase the complexity, and these need to be anticipated to enable smooth workflow operation.

Here are some typical questions and requirements that come up daily. Of course there are many more and ActiveFlow helps you meet them.

Real-world requirement	ActiveFlow
The form is almost correct, just a small change is required. I just want to return it to the maker for amendment and resubmission	✓
I submitted this form but realize the account code is wrong. I want to quickly cancel it, correct it, and resubmit it.	✓
This travel expense sheet can be submitted by any person in any department	✓
The travel expense sheet should go up the organization hierarchy for approval in the department in which it was submitted, before going to accounts for final approval.	✓
What has happened to my travel expense application?	✓
Let me see all the purchase orders approved in June by a certain person.	✓
For this form, to save time, I want all six of my managers to see it in parallel and comment on it before I make the final approval.	✓
If anyone rejects this form, the workflow item must be aborted and all signatories notified by email with the reason.	✓
How many card applications were received in July?	✓
My junior manager has not approved a document which must be approved immediately to meet the close of accounts deadline. The manager has gone down to the plant and cannot be contacted. We need to make an immediate emergency authorization for this one time.	✓
I travel frequently and need a quick way to assign a delegate for authorization.	✓
We frequently reorganize our company and want both the changes to the company structure and to authorization rights to come from the Personnel Department directly as a file.	✓
A manager returns from vacation and wants to know all work that has been authorized by the delegated approver.	✓
How long has the accounts manager been sitting on my claim form?	✓
How many more people have to approve my claim?	✓
I need to interface this form to an accounts database to see if there is sufficient budget before submitting to accounting.	✓

For tax purposes I need a list of all purchases approved during the year.	✓
How many application forms have we approved this month?	✓
I need to show this form to my deputy manager for her comments before I approve it.	✓
I forgot to delegate my authority before leaving for vacation.	✓
We want all organization and staff details, together with authorization rights, to come directly from the Personnel Department and to be loaded automatically to the workflow system	✓
We want details of new staff, retiring staff, and staff movements to come directly from the Personnel Department as an automatic file input to the workflow system	✓
I need to attach some rules for the routing. If the amount is greater that $10,000, the president wants to check it	✓

Now let's move on to some of the features of ActiveFlow.

Submitting a form

Submitting a form is easy. An authorized user sees a hierarchical list of workflow items that can be submitted. Here are a few examples:

Personnel Forms

- Travel Allowance
- Maternity Leave
- Change of Marital status
- Meal Allowance Claim.

Purchasing Forms

- Purchase request for an Expensed Item
- Purchase request for a Fixed Asset.

The user just fills in the form and presses the Submit button. The workflow designer has created these forms, with the appropriate validation logic, using any form editor.

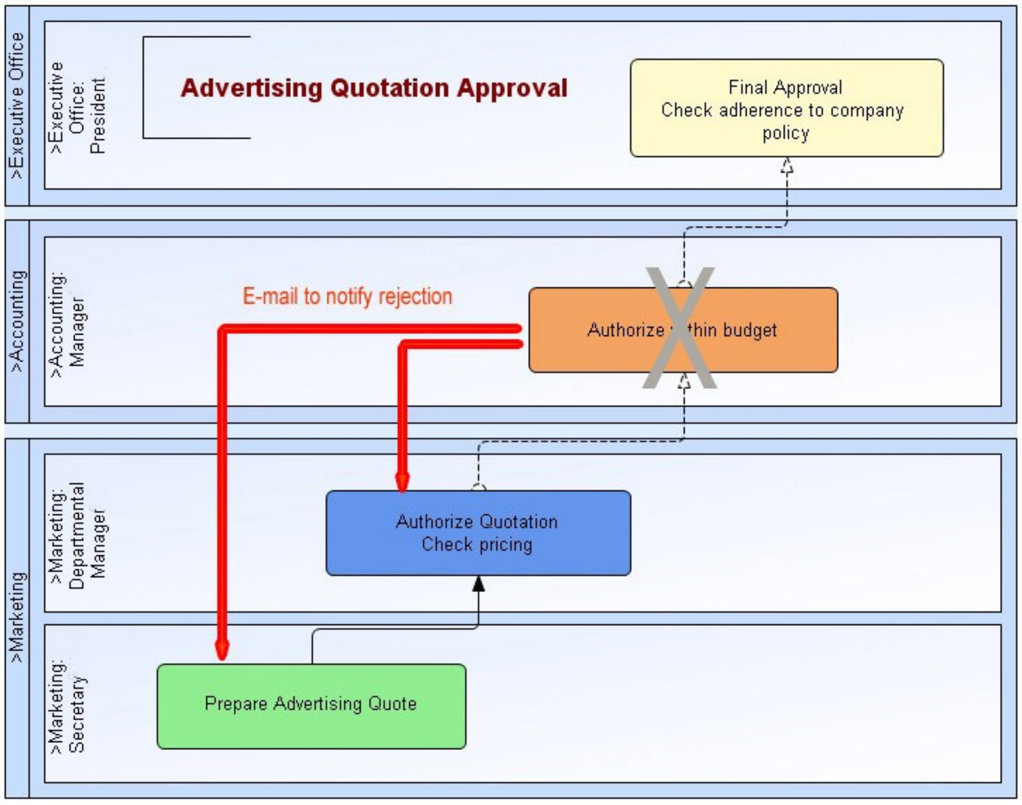
Normal approval

A user accesses ActiveFlow, checks the In-tray, and selects a form. If the user agrees with the contents, he/she presses the Accept button. That is all the user has to do. ActiveFlow automatically routes the form to the next stage of the authorization chain.

Rejection

A form has been sent to a user who decides not to authorize it. Some examples of rejection reasons could be: the product is too expensive, there is not enough budget, the applicant has already exceeded the allotted number of holidays.

The form designer simply includes a Reject button in the form to give the option of Rejection (as well as the approval button).


>Executive Office
>Executive Office: President
Advertising Quotation Approval
Final Approval
Check adherence to company policy
>Accounting
>Accounting: Manager
E-mail to notify rejection
Authorize within budget
>Marketing
>Marketing: Departmental Manager
Authorize Quotation
Check pricing
>Marketing: Secretary
Prepare Advertising Quote

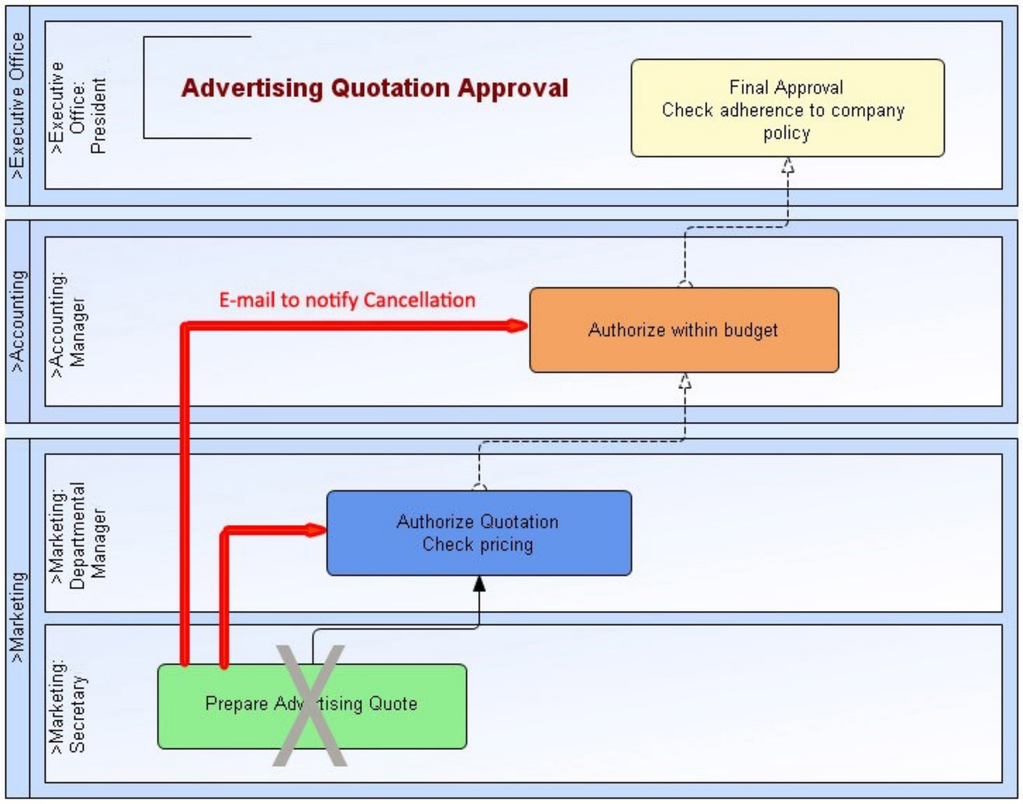
In the above example, the Accounting Manager has rejected the advertising quotation because the Marketing Department does not have sufficient budget to cover the cost.

ActiveFlow ends the workflow and sends an email to notify all participants who have signed the form, including the maker. A reason for rejection is required with ActiveFlow and the reason given by the Accounting Manager is included in the email. The form is archived and an audit trail record is also created regarding this event.

Cancel workflow

Only the maker can cancel a form and it must be done before final approval has been granted. In such a case the maker has submitted a form and a number of authorizers could have signed it already.

The maker decides there is something wrong with the form (perhaps a wrong account code was used) and can cancel the form before the final approval.


>Executive Office
>Executive Office: President
Advertising Quotation Approval
Final Approval Check adherence to company policy
>Accounting
>Accounting: Manager
E-mail to notify Cancellation
Authorize within budget
>Marketing
>Marketing: Departmental Manager
Authorize Quotation Check pricing
>Marketing: Secretary
Prepare Advertising Quote
X

In the above example, the Maker cancels the form before the President can sign the advertising quotation.

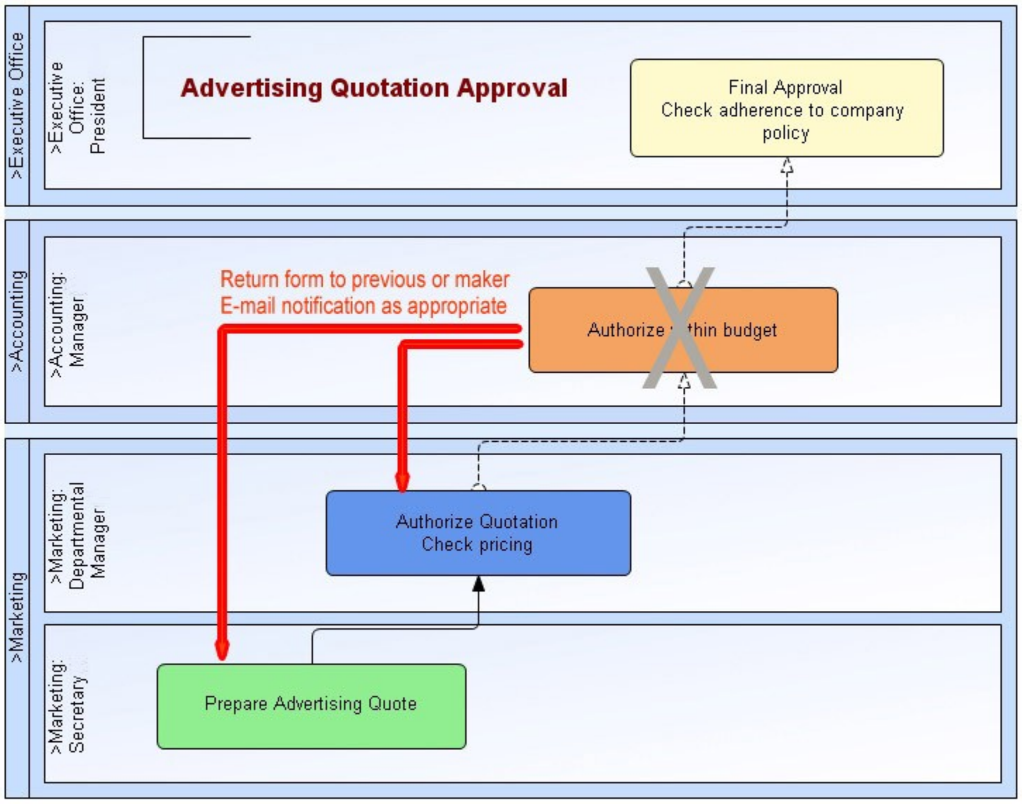
ActiveFlow ends the workflow and sends an email to notify all participants who have signed the form, including the maker. A reason for cancellation is required by ActiveFlow and the reason given by the Maker is included in the email. An audit trail is maintained regarding this cancellation.

Return

This is a useful soft form of reject. Rather than rejecting a form outright, it is returned to either the previous signer or the maker, so that they can change the form content and submit it again.

For example, an authorizer notices a small error in a form: the maker gave a wrong account code. So the authorizer returns the form to the maker. In the case of a workflow adding content step-by-step, a return to previous could also be used and the previous authorizer would change and resubmit the workflow. An audit trail is of course maintained regarding any return actions.

The form designer would include a **Return** button in the form to give the option to **Return a** form.


>Executive Office
>Executive Office: President
Advertising Quotation Approval
Final Approval Check adherence to company policy
>Accounting
>Accounting: Manager
Return form to previous or maker
E-mail notification as appropriate
Authorize within budget
>Marketing
>Marketing: Departmental Manager
Authorize Quotation Check pricing
>Marketing: Secretary
Prepare Advertising Quote

In the above example, the Accounting Manager decides to return the form and has two choices: return either to the previous authorizer or to the maker.

ActiveFlow returns the form to the previous authorizer or to the maker and sends an email to notify all participants who have already signed the form (including the maker for the return to maker case). A reason for return is required by ActiveFlow and the reason given is included in the email. An audit trail is maintained regarding this cancellation.

Delegate approver

An authorized signer could be away from the office and wishes to delegate the signing responsibility to somebody else.

The user logs in to ActiveFlow, then in the administration area chooses **Delegation**. An organization tree control appears, and the user can choose who will be the delegate and optionally set dates for the delegation as well . This can be someone in a higher or lower position, and not necessarily in the same department. All work will then be routed to the delegate.

If a user forgets to set the delegation before leaving the office, it can be set up by a user who has Administrator rights.

While this delegation is in force, an email will be sent to notify the original user about all work items authorized by the delegate. When the user returns, it is easy to see all the work has been authorized under delegation. The user can switch off the delegation, again in the administration forms area of ActiveFlow if specific dates for delegation have not been used.

Users who are frequently away from the office and want to nominate the same delegate can keep that delegate as an assignment candidate and switch the delegation on and off.

Delegate maker

A department manager or the president of the company often doesn't want to spend time entering workflow items such as travel expense details and can delegate a secretary or subordinate to do these tasks. The delegate maker will have the same starting rights as the original user.

The form raised by the delegate maker will be sent to the original user for checking and after that it will follow the logic described in the map.

If a user in the approval chain returns the form to the maker the delegate maker will receive it.

Copy form

Often when starting a workflow, it is useful for a maker to refer to and copy the contents of a previous workflow to save time and check for content. For example, a sales person who visits the same customer each month will have similar travel expenses and can copy the previous expense claim as a base. Another example would be when a maintenance purchase requisition has to be completed. Here the maker could refer to the last form (e.g. last year) and copy the content as a base for the new application.

Any user can start a new workflow with data from a previously sent/approved workflow.

Hold form

A form partially filled with data can be put aside in the Hold tray to finish it later. An example here is when a sales person enters data in a travel expense form daily but will submit it only once every 2-3 days.
Another example is when a maker is called away to an urgent meeting and holds the form for later completion.

Not only the maker can hold a form. A form can be moved by an approver from his normal in-tray to the hold tray.

Emergency action

This is a useful feature in the real world of workflow.
As an example, your junior manager has not approved a form that must be approved immediately to meet the close of accounts deadline. She has gone to the plant and cannot be

contacted. You need to make an immediate emergency authorization for this one time. So you log in to ActiveModeler and choose **Emergency action** from the **Special Functions** menu. You can view the In-tray for the selected user and choose which work item you want to authorize.

With the **Emergency action** function you can select the In-tray of users within the same department and for one level below the logged-in user. An email is sent to the original user to explain what happened while they were absent.

Bubble-up

When you define a workflow it may not be convenient to define all workflow paths explicitly. A holiday application form, for example, can be filled in by any member of staff in any department.

Defining all departments and all roles would be a long job. ActiveFlow works as follows: for a specific department or a department defined as a General Department, only one activity is defined in ActiveModeler. But a **bubble-up option** is added to it (right-click on the task to define this option).

The bubble-up option means that starting from the maker, the authorization path will follow the chain of hierarchy automatically within the department. That is the default or primary path. An optional secondary bubble-up path can be set because some workflows might need a different workflow chain in a department. For example, holiday forms might be handled differently from expense forms.

The hierarchy itself (who is the boss of whom) is handled in the candidate database.

Normally a workflow would bubble-up in the maker department, then go on a fixed route (such as from the maker department to accounting, then to the financial controller), but ActiveFlow allows multiple bubble-up processes in a process map.

Searching for a form

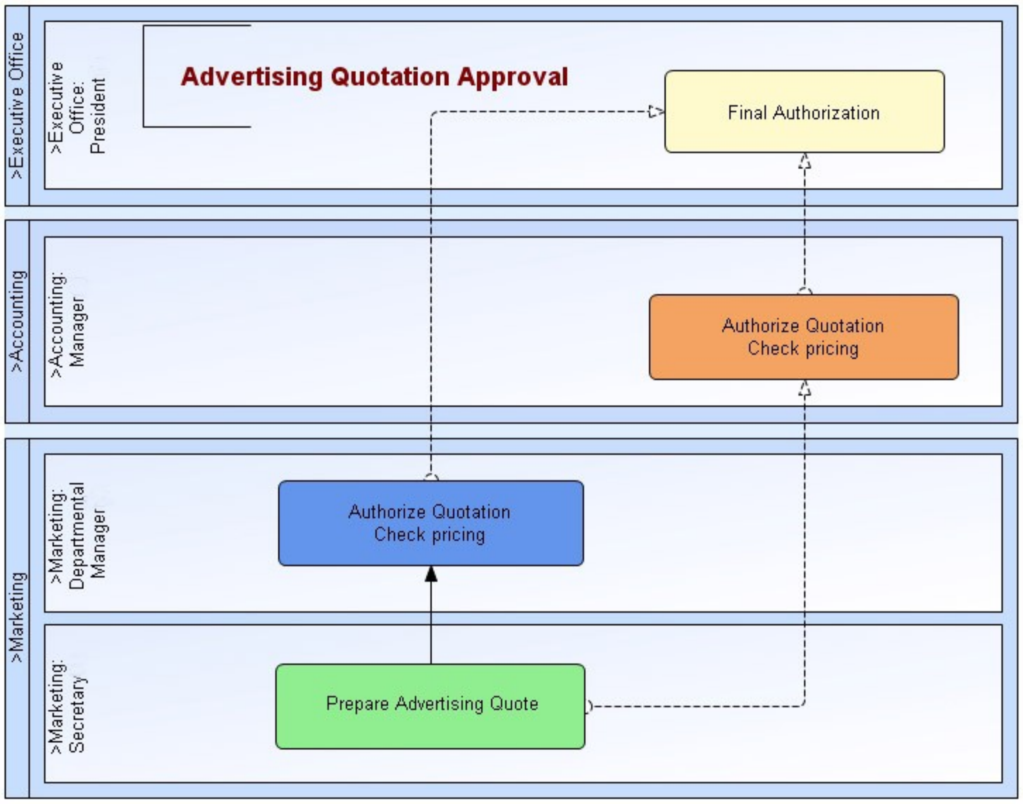
It is important to be able to track each form to check who has it, whether it has been approved, and so on.

ActiveFlow offers the following views:

- Trace workflows issued/approved by somebody or within certain department
- Forms waiting for approval
- Pending forms (Approver view)
- Approved forms.

Also, a financial controller should be able to see **"everything"** and for this yjr controller can use the Special View inquiry page.

This brings us to the difference between **Forms waiting for approval** and **Pending forms**.


>Executive Office
>Executive Office: President
Advertising Quotation Approval
Final Authorization
>Accounting
>Accounting: Manager
Authorize Quotation Check pricing
>Marketing
>Marketing: Departmental Manager
Authorize Quotation Check pricing
>Marketing: Secretary
Prepare Advertising Quote

In the case of a parallel route as above, the President will not receive a form for approval until both the Accounting Manager and the Marketing Department Manager have authorized it. But the President can use the **Pending form search** function to see the form - who has signed it, and who has yet to sign it. (Another example could be for say all six engineering managers in a manufacturing plant needing to sign off a new drawing change as part of a parallel route).

Back to this case again and after everyone has approved, the President receives the form as a **Form waiting for approval**. Note that a simple workflow chain has no **Pending forms**, just those waiting for approval.

Where to use ActiveFlow

The range of both the industries and the applications themselves is not limited in any way. In fact we are always being surprised by the ways in which the combination of ActiveModeler and ActiveFlow is being used by customers.

Here are just a few of the areas in which ActiveFlow can be used:

Use	Industry
Application forms (including intranet sources)	All
Call Centre Tracking	Call centers
Capital Expenditure Authorization	All
Engineering Change Requests	Engineering
Engineering Status Tracking	Engineering
Expense Forms	All
Financial Performance Consolidation	All
Information Distribution and Feedback Analysis	All
Patient Tracking	Hospitals
Personnel Forms - such as Holiday, Marriage, Address change, Resignation, New Employees, and Travel allowance.	All
Problem Reports	Engineering, Call centres, maintenance centres
Product Life Cycle Development	All
Purchase Order Authorization	All
Software Development Life Cycle (SDLC)	Software Engineering
Suggestions Forms	All
Surveys (including intranet source)	All
Telephone Centre Control	Banking, Call Centres
Time Sheets	All
Asset Tracking	All
Office Automation	All
Shipment Tracking	All
Proposal Tracking	All

Steps to making a workflow

How are workflows developed? Usually there is already a business process being performed manually with a paper-based flow. A workflow aims to replace the paper, the manual controls, and human thought processes with an automatic and reproducible system.

The entire manual process can be reproduced to form the new workflow or it can be rationalized to produce an optimal flow. This is where the linkage to our strong ActiveModeler modeler product becomes a real asset. Many other workflow products have a weak modeling component; unlike the strong coupling of ActiveModeler and ActiveFlow.

Model a business function

To model a business function we use our ActiveModeler Avantage product. The ActiveModeler User's Guide describes the functionality in detail. Functions related exclusively to workflow are described in this Design Guide.

As a workflow example, let's examine a Purchase Requisition workflow. These are common in most companies.

Purchase Requisition example

Brief

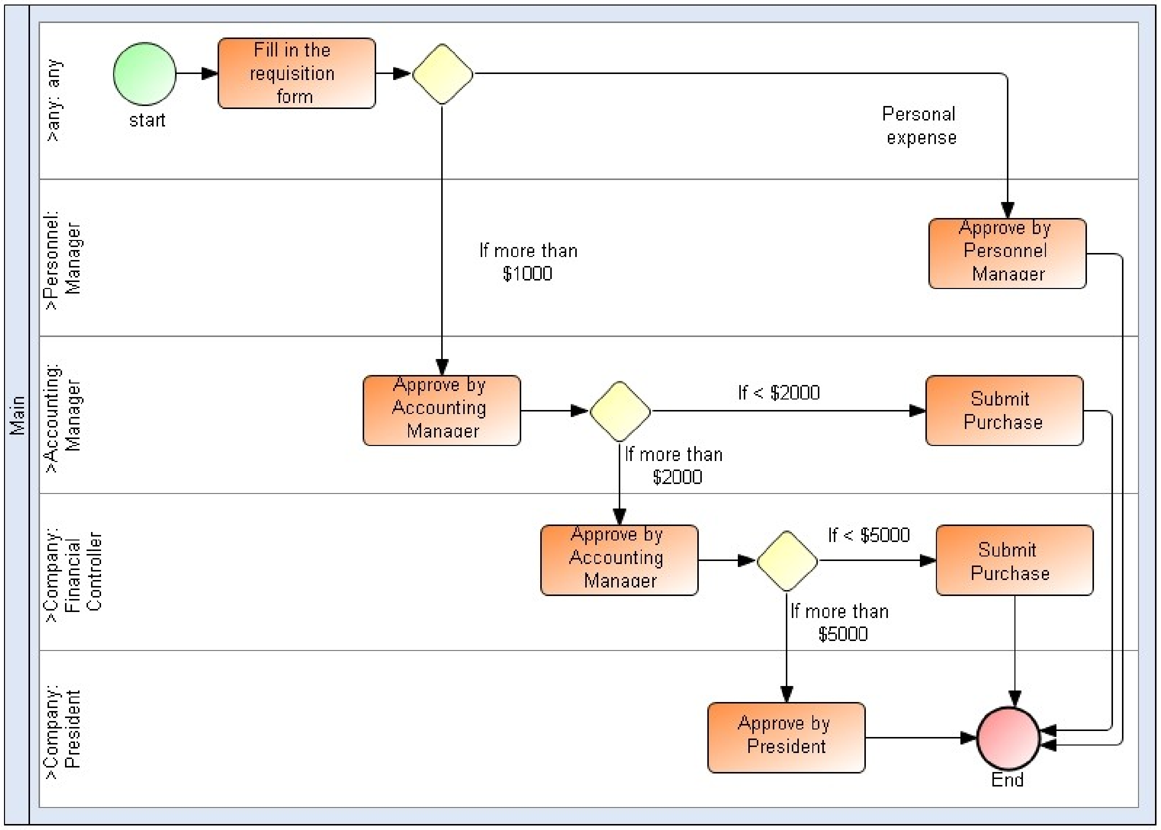
A company needs rapid deployment of an automated Purchase Requisition workflow to replace their current paper-based system. The big problem with the existing system is the tracking of paper forms (knowing who has got what) and the time taken to obtain approval from all parties To make things worse, there are six branch offices remote from the main office, and the paper forms have to be sent around by normal inter-office mail.

High-level requirements

- A Purchase request can be made in any department in the company (including the branch offices) by an authorized maker
- The system must check that budget is available within the department against the account code of the request item. If budget is not available, reject the request at the maker level
- The Purchase request should bubble-up through the maker department to the departmental manager level for authorization.

The following **routing rules** dictate who should approve the expense:

- If a personnel-related expense, send to the Personnel Manager
- If more than $1000, send to the Accounting Manager
- If more than $2000, send to the Financial Controller
- If more than $5000, send to the President.



You can see the completed map above. The submission process is as follows:

- A maker in any department can submit a Purchase Requisition request form
- The workflow checks the Accounting database to make sure sufficient funds are available in the department for the required item or service
- If sufficient funds are not available the form cannot be submitted

When the form is successfully submitted by the maker, the form bubbles-up through the hierarchy of the maker department. For your information - for flexibility, there is a choice of two routings: a primary and a secondary route. This means that for some forms a special bubble-up route could take place (for example only including a certain technical manager in some circumstances).
After the bubble-up is complete, the form follows the predefined route shown on the map.

In our example the standard primary route will be adopted.

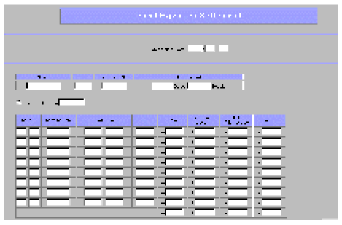
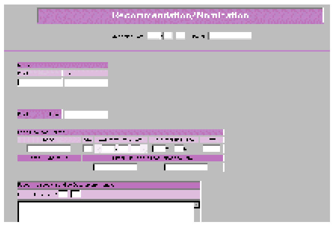
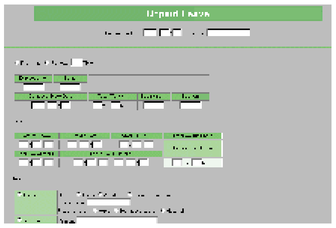
Design the forms

Forms enable work to be passed between users. ActiveFlow forms are designed with any HTML or WebForms (ASP.NET) editor . In this way, we use a standard and commonly used design tool rather than our own proprietary forms designer.

In ActiveModeler Avantage set the workflow form type (ASP or ASP.NET) in the diagram settings dialog.

The starting point for the design of a workflow form will often be a paper-based form which is already in use. Taking this form as a base we can transform it into an attractive electronic form.

For example, the forms below were originally on paper in a set of personnel forms.

After designing, the forms must be defined to ActiveModeler. To do this, go to the workflow process map and right-click on the activity where the form is used. A menu appears. Choose the **Settings...** under the **ActiveFlow** menu and then select the html or aspx file. Those are the required steps, but you may also want to define or code special business logic as described below.

Add the business logic

The business flow is defined by the map, but there are usually additional requirements for the business logic as follows:

Basic Form Validation

The form designer does this in the form editor , building-in date ranges, numeric checks, range validation, and so on.

Process Logic

Checking for certain values in the form with a resulting event. The example above had four conditions to be checked for:

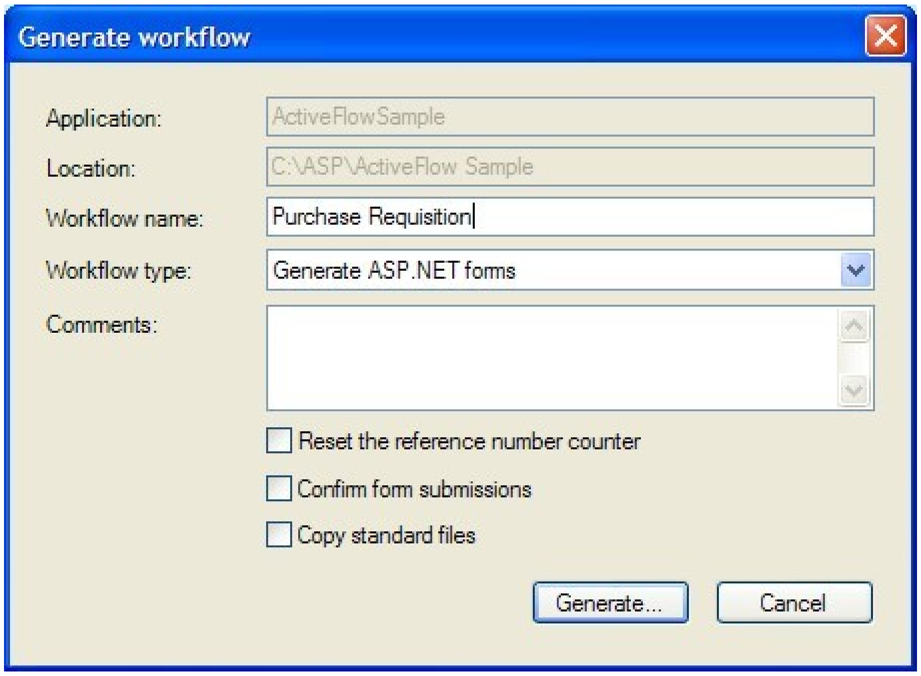
- If a Personnel-related expense, send to the Personnel Manager
- If more than $1000, send to the Accounting Manager
- If more than $2000, send to the Financial Controller
- If more than $5000 send to the President for approval.

The code for these checks would be automatically generated by the rules wizard.
This logic would be added in ActiveModeler for the Accounts Manager Activity. The workflow designer would right-click on the activity shape in the modeler, choose **ActiveFlow->Rules...**, and add the logic in the Workflow rules **Transition Condition** section. In this way, even complicated logic can be catered for, in a simple way.

Run the Workflow Wizard

After the designer has completed work on the forms, defined them to ActiveModeler and written the business logic in the Workflow rules, it is time to run the Workflow Wizard to create the Workflow.

From the ActiveModeler map, the designer right-clicks the map itself and selects **Compile...** under **ActiveFlow** menu. The Workflow Wizard dialog box shown below appears. The designer completes the required information about the workflow to be generated and presses the **Generate** button. ActiveModeler generates the complete workflow including validation and business logic requirements.



Generate workflow

Application: ActiveFlowSample

Location: C:\ASP\ActiveFlow Sample

Workflow name: Purchase Requisition

Workflow type: Generate ASP.NET forms

Comments:

☐ Reset the reference number counter

☐ Confirm form submissions

☐ Copy standard files

Test the system

As with any system, an ActiveFlow workflow must be tested properly before it becomes a production system. A test plan needs to be produced to test all functions and validation logic. The test environment should be the same, from logical point of view (organization structure, database type, external connectivity to other systems), as the production environment. When the problems (if any) have been corrected on the test environment and the designer is happy with the workflow, is time to introduce it in production.

Introduce the system

After the system has been thoroughly tested, it should be ready for production. It is good practice to try out a system in a limited pilot at first, perhaps for one department rather than for the whole company. Any issues will be smaller and easier to control and enhancements can be made before introducing to the whole company.

The pilot will effectively be an extension of the server test. The steps to introduction of the pilot are:

- Copy the map to the production server
- Run the Workflow Wizard on the production server (if not already done)
- Add users to ActiveFlow through the **Add user** Admin function (if these users are not present from previously defined workflows)
- Add candidates for the workflow (who will participate in the workflow), via ActiveModeler.

After a successful pilot the workflow can be put into full production.

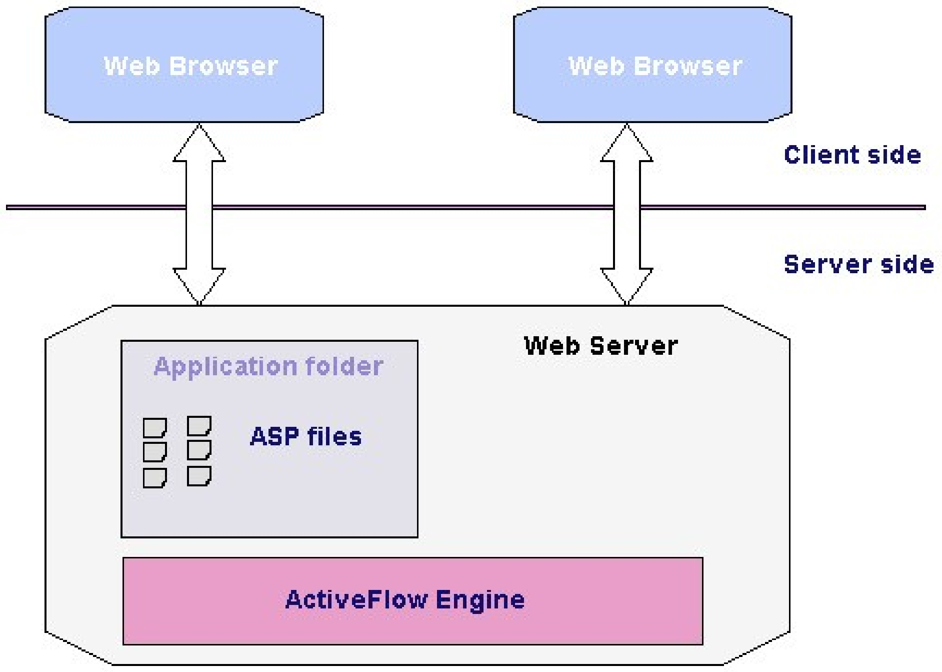
We hope your Workflow succeeds in making your business more efficient!

ActiveFlow workflow design

Introduction

ActiveModeler Avantage provides an environment where you can examine business processes, as they are, right down to the finest detail and then later devise improved or automated process.

ActiveFlow is a "web-based" application. End-users need only a web browser in order to submit/handle workflows. On the server side, the web server receives requests from users and the ActiveFlow engine fulfills these requests.

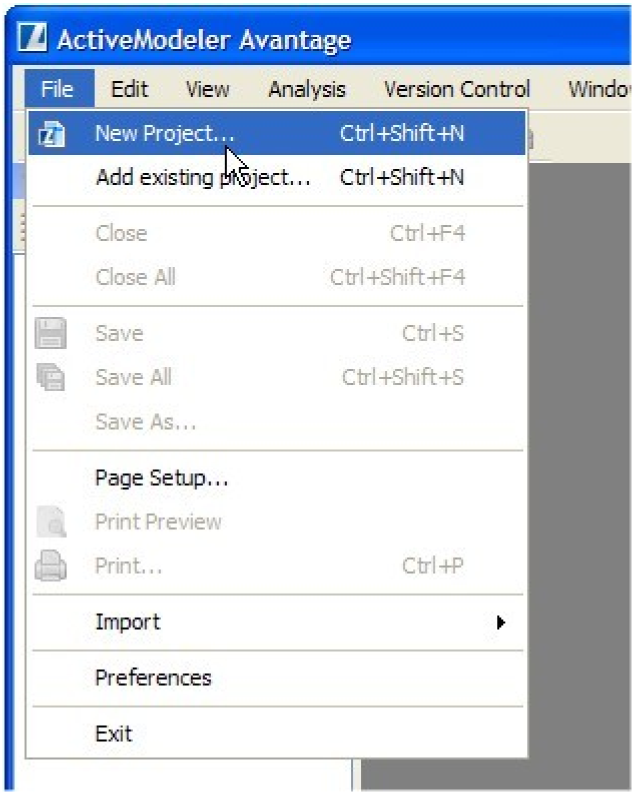


Below are presented the terms and the concepts used for creating a workflow.

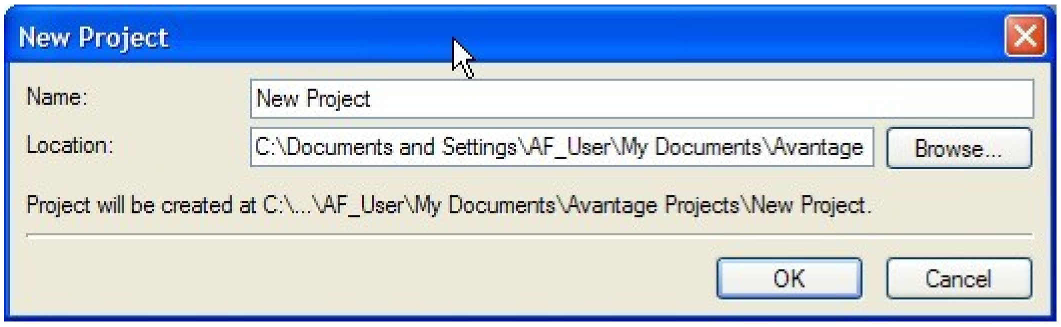
ActiveModeler Avantage project

With ActiveModeler all information is stored in the project repository. So the set-up of this repository is the first action the designer should do before starting to develop workflows. It is necessary to do this only once.

To create a new project select **File\New project** option from the Avantage menu.



You will have to specify the name of the project and the location.



Name: New Project

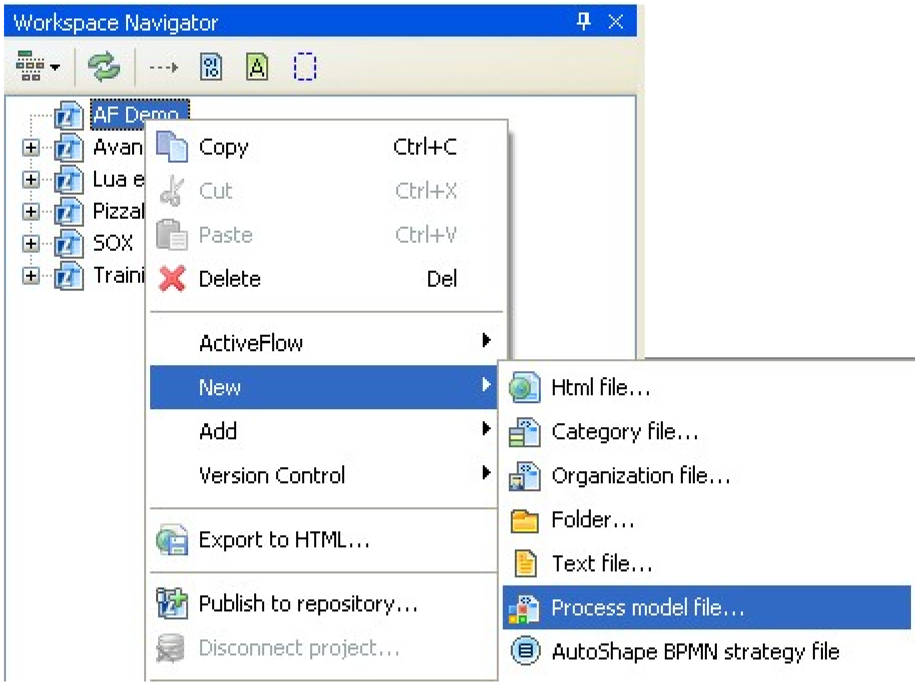
Location: C:\Documents and Settings\AF_User\My Documents\Avantage

Project will be created at C:\...\AF_User\My Documents\Avantage Projects\New Project.

ActiveModeler Avantage model file

The model file is used by Avantage to group and store information about the logical or physical entities which are to be modeled. An Avantage project may contain several model files. For example, a project file can represent the entire company and the models can group the business processes from different departments: Sales, Marketing, Production etc.

To create a new model file, right click on the project item and select **New\Process model file...**

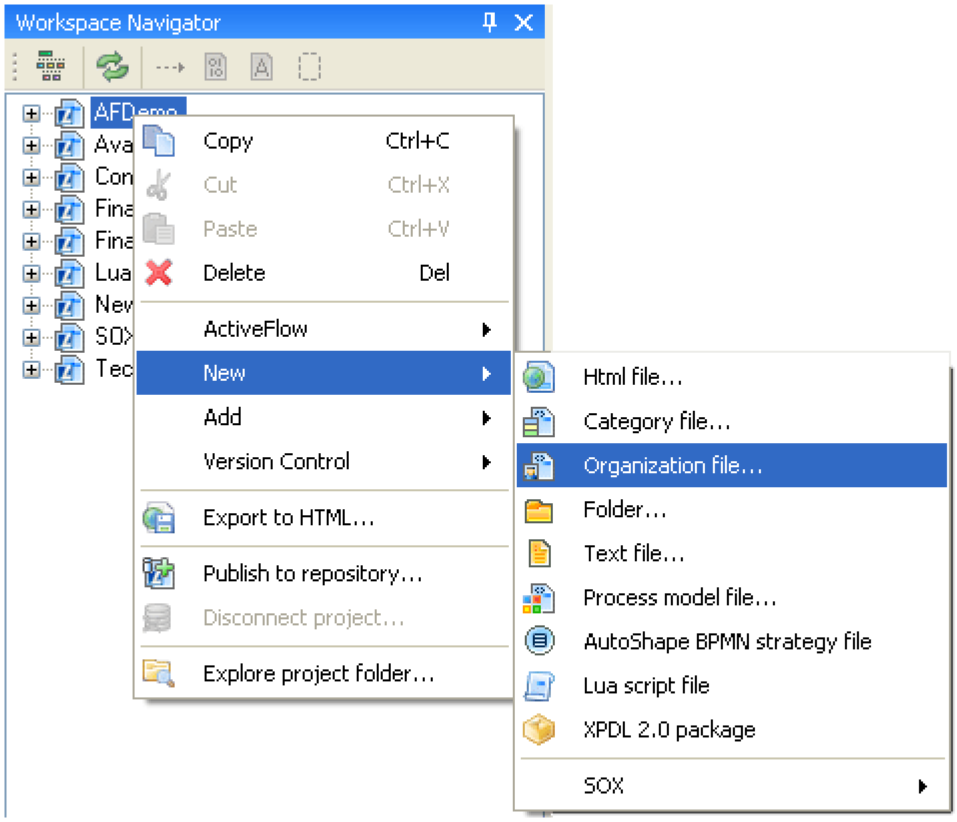


Organization file

When the project has been created it is necessary to define the organization structure, i.e. the departments, roles and the users.

The organization structure is stored in 2 different locations: a local file on the disk and the ActiveFlow database. You have to make sure the data stored in the 2 location is always synchronized.

- **Creating a new organization file for the project**



To create a new organization structure file right click on the project item and select **New\Organization file.**
To add new organization items such as departments and roles right click on the organization item and select **New...** and then select the desired item type: department or role. You may have any number of departments, sub-departments and roles on any number of sub-levels for departments.

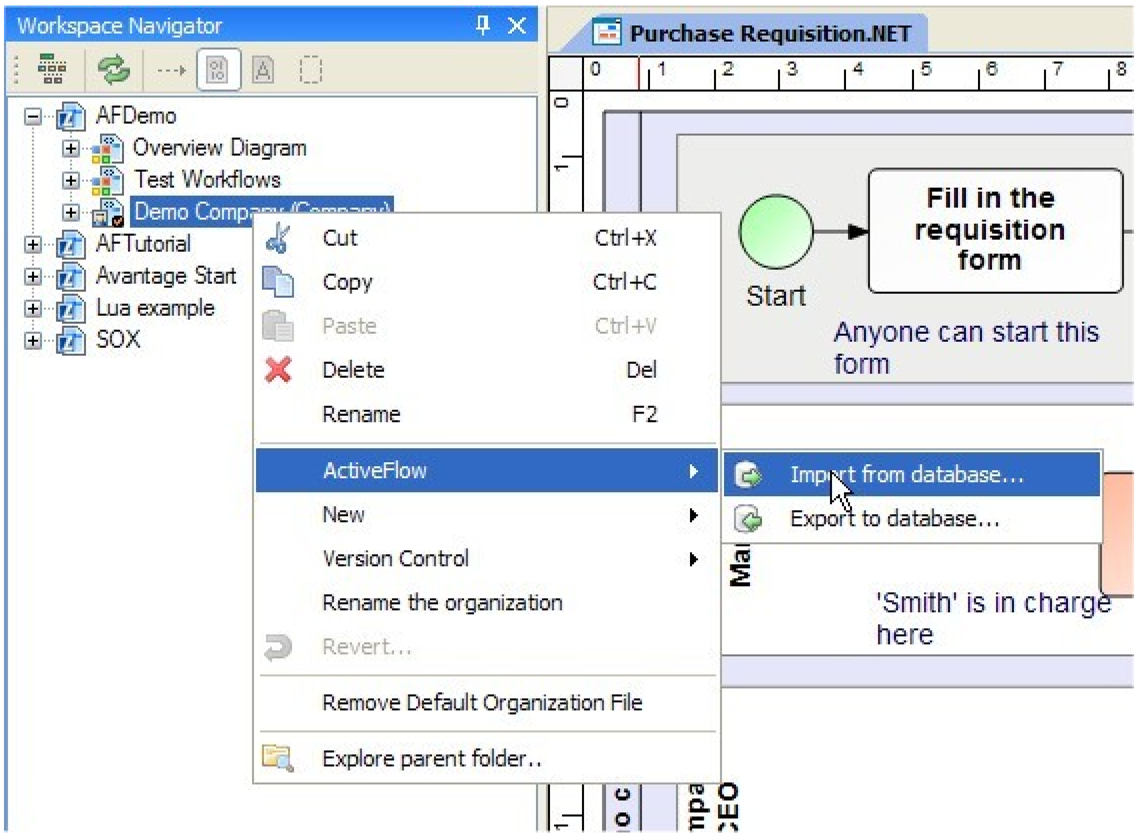
The organization structure is stored in a file with the extension .orgstructure .
Now you can easily copy organization files across projects or share organization structure with other ActiveModeler Avantage users. Another useful feature is the possibility to have multiple organization files and switch between them by changing the default .

- **Importing/exporting the organization structure from an existing ActiveFlow database**

Importing the organization structure from an ActiveFlow database will erase all the existing items in the **.orgstructure** file and will add all the items existing in the specified ActiveFlow database.

Note:
After creating the .orgstructure file, the department and roles can be also imported from an existing ActiveFlow database. This si useful when you already have the organization structure

defined in a different ActiveFlow project, or you already use previous version of ActiveModeler/ActiveFlow.



Exporting behaves in the same fashion in the opposite direction, deleting all the organization items in the ActiveFlow database and inserting the items existing in the current .orgstructure file.

Note:

It is **NOT** recommended to change the organization structure (add departments and roles) from the **ActiveFlow -> Administration** menu. Use the ActiveModeler Avantage instead and then export the changes to the ActiveFlow database.
ActiveFlow users cannot be added using ActiveModeler. This is because ActiveModeler is primarily a business modeling tool. In order to define the users you have to use the Batch Admin Toolset or the administrative pages in ActiveFlow as described in the ActiveFlow User Guide.

Diagrams

The diagrams model the business processes using the events, tasks, gateways and links.

In order to create a new diagram right click on the **Automated** item under the model and select **New\Diagram**.

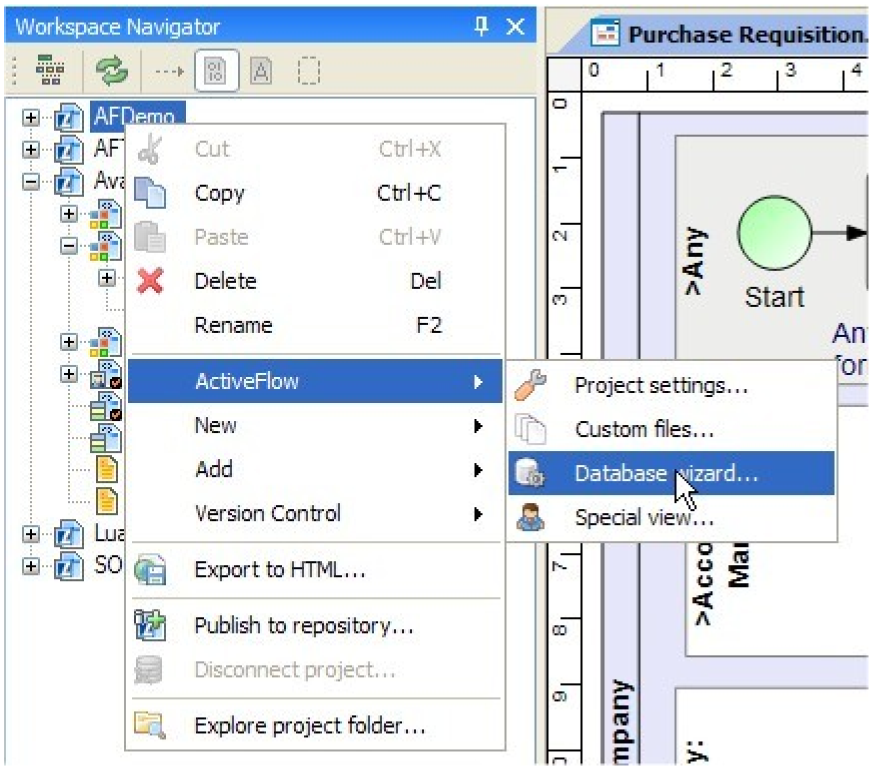
Setting up ActiveFlow database

There are two possibilities when setting up the ActiveFlow database :

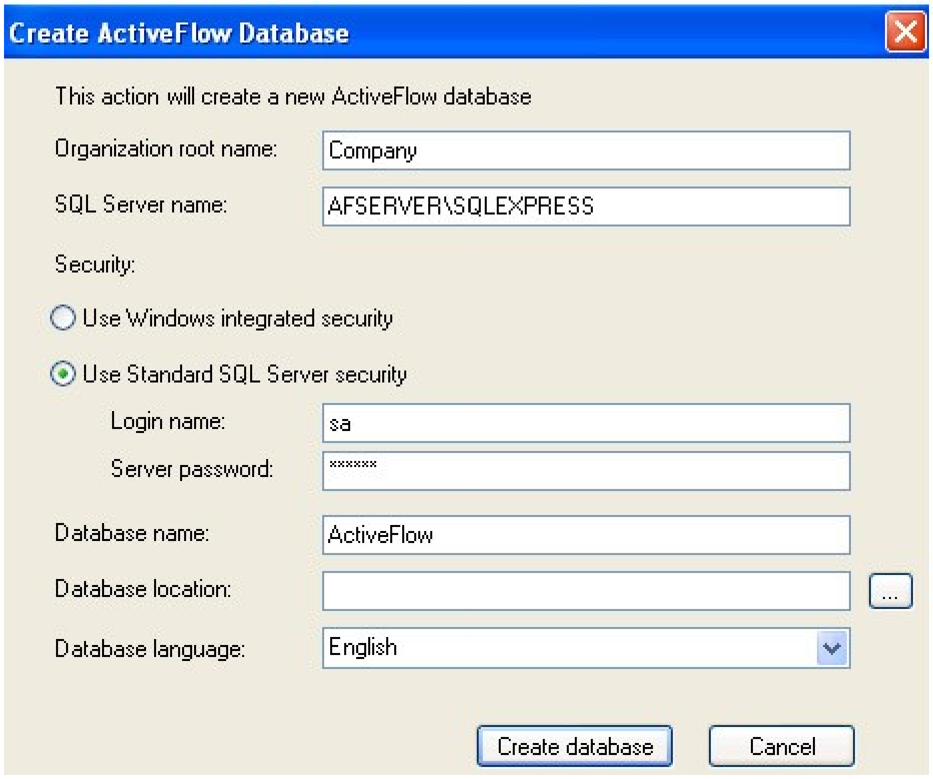
- **Create a brand new ActiveFlow database:** This option will be normally used by the first person to create an Avantage project for ActiveFlow. See the next slide for details.
- **Link the project to an existing database.** This operation might be useful for people using a project already created that needs to re-establish the database connection settings and must be re-linked to the ActiveFlow database that resides on the ActiveFlow server. The necessary settings are described in one of the next slides: "Edit the ActiveFlow project settings- Database tab"

Creating a new database

- Right-click the project item then select **ActiveFlow -> Database wizard..**



- Complete the required fields and press the **Create database** button



This action will create a new ActiveFlow database

Organization root name: Company

SQL Server name: AFSERVER\SQLEXPRESS

Security:

○ Use Windows integrated security

● Use Standard SQL Server security

Login name: sa

Server password: ******

Database name: ActiveFlow

Database location:

Database language: English

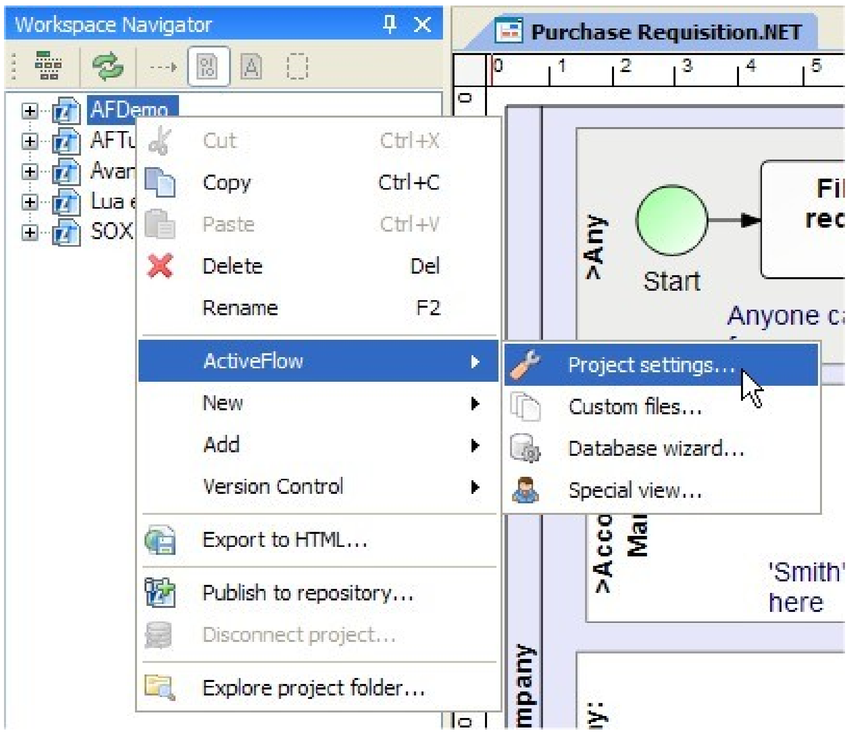
Notes:
- A progress dialog occurs while the database wizard is running. The progress dialog might be hidden behind the Avantage window but you can find it in the task bar.
- Depending on the environment configuration, while the DB wizard is running you might get some warnings or non-critical error messages. You can safely ignore these and press OK to resume.
- The **Database location** must be the local path where the database files are saved on the SQL server machine. Ask the server administrator for details.

Project settings

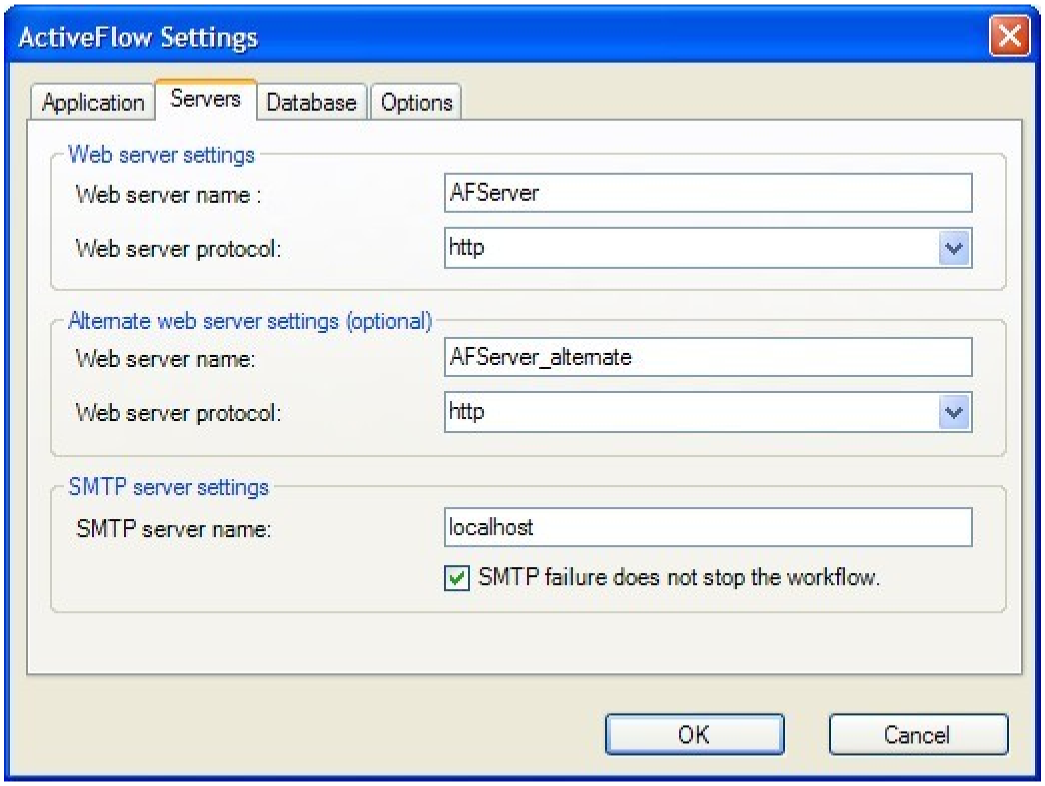
ActiveFlow is a web application so it is necessary to specify a web server. Also it requires (optional) access to a mail server for sending mail notifications in certain situations.

Note: These values have to be set only once per project.

Right-click the project item and select **ActiveFlow**-> **Project** S**ettings**..



SERVERS TAB



Web server settings

Web server name : AFServer

Web server protocol: http

Alternate web server settings (optional)

Web server name: AFServer_alternate

Web server protocol: http

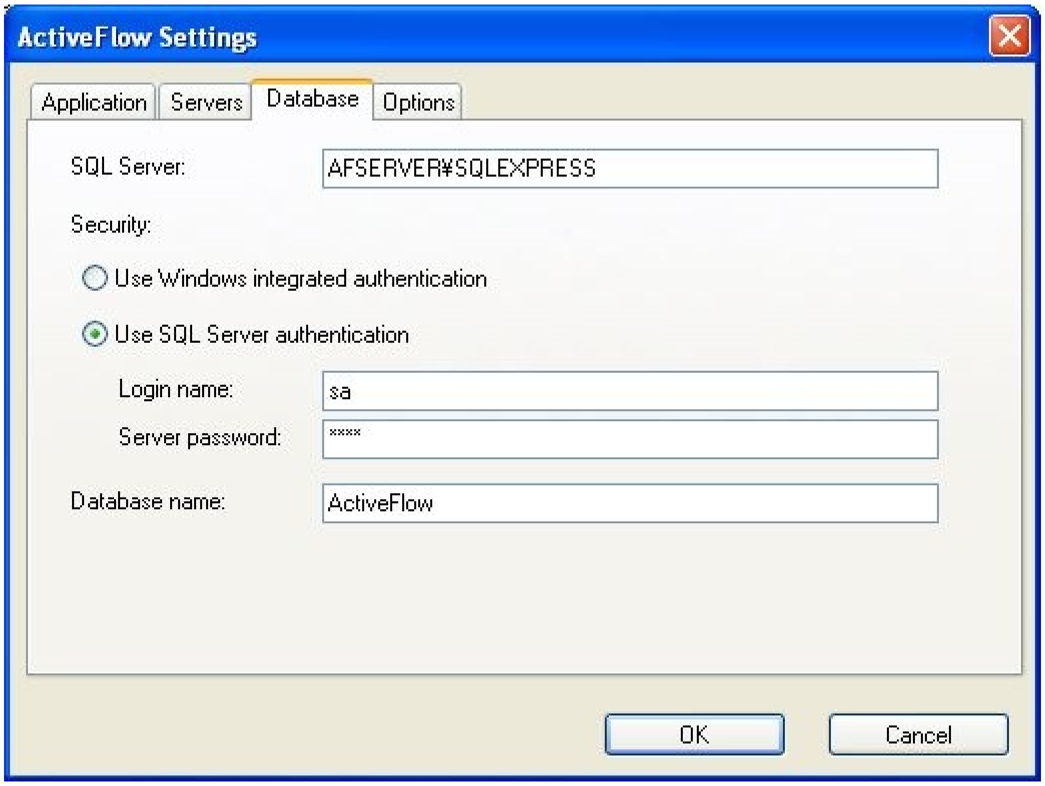
SMTP server settings

SMTP server name: localhost

[✓] SMTP failure does not stop the workflow.

Web server settings	
Web server name	The name of the web site where the ActiveFlow application is registered.
Web server protocol	The protocol used to access the web site where the Active Flow application is registred (HTTP or HTTPS).
Alternate web server settings (optional)	
Web server name	The name of an alternate web site where the ActiveFlow application is registered.(if applicable)
Web server protocol	The protocol used to access the alternate web site where the Active Flow application is registred (HTTP or HTTPS).
SMTP server settings	
SMTP server name	The name of the SMTP mail server which can be used by ActiveFlow to send the e-mail notifications (optional).

DATABASE TAB



SQL Server: AFSERVER¥SQLEXPRESS

Security:

Use Windows integrated authentication [✓ Use SQL Server authentication]

Login name: sa

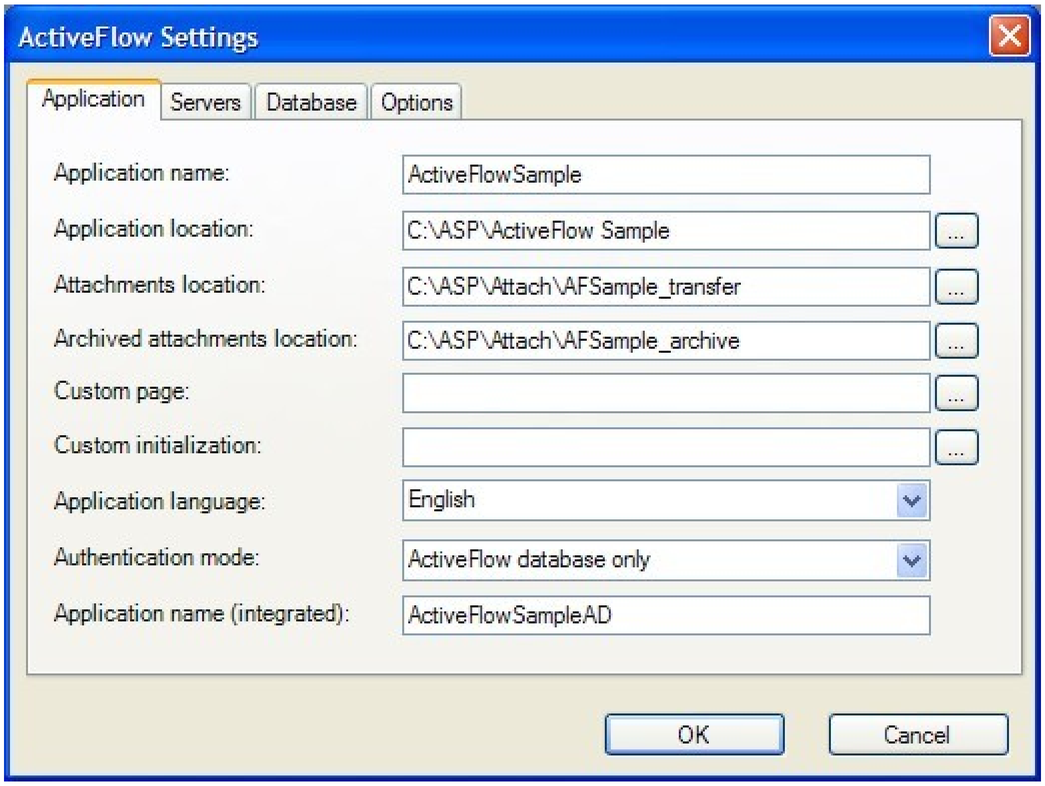
Server password: ****

Database name: ActiveFlow

The database settings are explained below:

Server name	The name of the MS SQL Server installed on the AFServer machine
Login name	The name of the SQL user who has administrative rights (sa)
Password	The login password of the SQL user

APPLICATION TAB



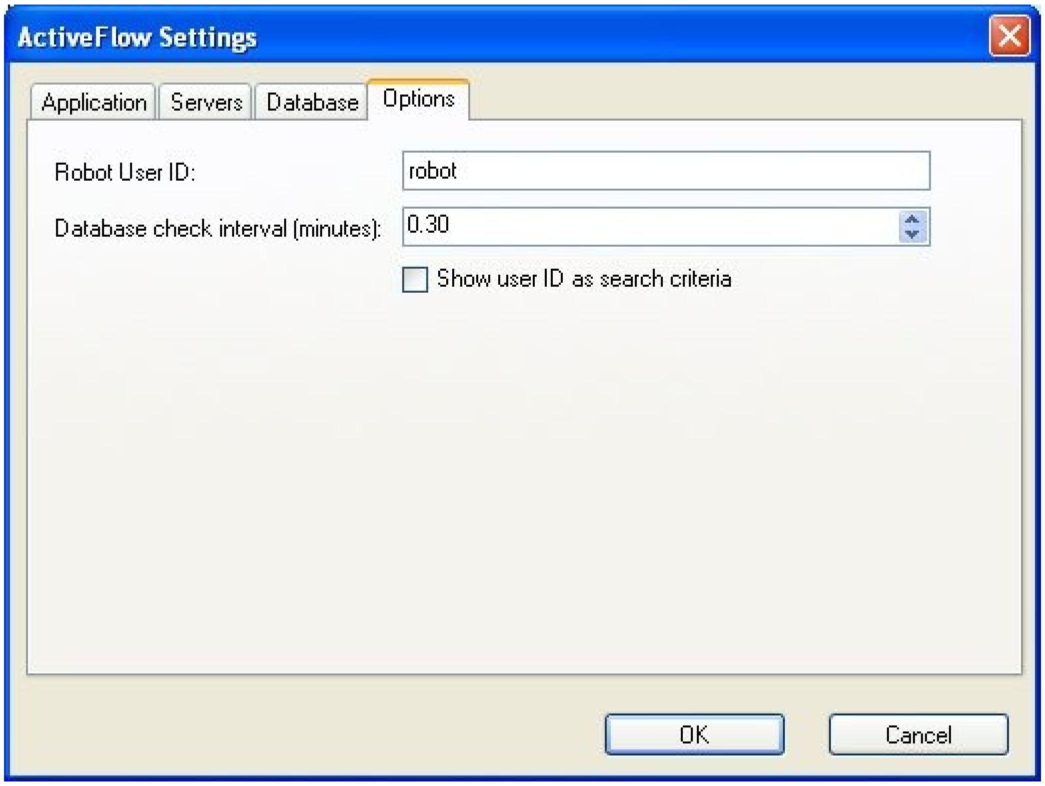
Application name:	ActiveFlowSample
Application location:	C:\ASP\ActiveFlow Sample
Attachments location:	C:\ASP\Attach\AFSample_transfer
Archived attachments location:	C:\ASP\Attach\AFSample_archive
Custom page:	
Custom initialization:	
Application language:	English
Authentication mode:	ActiveFlow database only
Application name (integrated):	ActiveFlowSampleAD

The application settings are explained below:

Application name	The name of the ActiveFlow web directory on the server
Application location	The network path of the folder on the server where the ActiveFlow files will be generated.
Attachments location	The local path on the ActiveFlow server that is used for attachments of the in progress workflows. Ask the server administrator if you don't know the local path on the server.
Archived attachments location	The local path on the ActiveFlow server that will be used for archiving files attached to archived workflows.

Custom page	The location of the custom page, the page which can be used by the workflow designer for linking to other applications.
Custom initialization	The location of the file where the designer can define custom global variables, global functions etc. This variables and functions will be visible from the workflow rules. (only for ASP forms)
Language	Specify the ActiveFlow default language. The language used for displaying the Login page etc.
Authentication mode	Specify whether to use integrated authentication (ActiveDirectory) or not.
Name (integrated security)	The name of the ActiveFlow web directory on the server, used for integrated authentication (optional).

OPTIONS TAB



Robot User ID: robot

Database check interval (minutes): 0.30

☐ Show user ID as search criteria

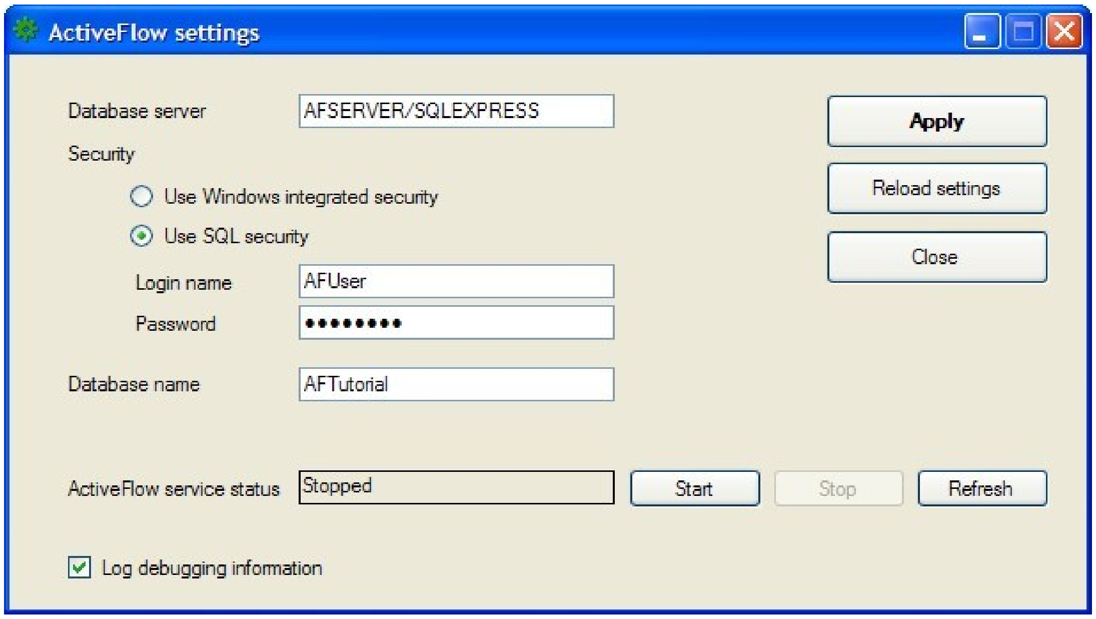
The optional values are described in the table below:

Robot user ID	The name of the ActiveFlow robot user used for automatic forms handling.

Database check interval	Specifies how often the ActiveFlow engine should check the database for expired forms or for forms which have to be automatically handled.
Show user ID as search criteria	If selected, allows the ActiveFlow user to use the userID as search criteria instead of selecting the full user name.

Setting up the ActiveFlow engine

Having setup the ActiveFlow database, it is time to configure the ActiveFlow engine. To do this we have to do use an utility application located on Start ->Programs -> ActiveFlow->ActiveFlow Service Settings. This is a small application used for controlling the state and settings of the ActiveFlow server.



Database server: AFSERVER/SQLEXPRESS

Security

( ) Use Windows integrated security

(●) Use SQL security

Login name: AFUser

Password: ••••••••

Database name: AFTutorial

ActiveFlow service status: Stopped

☑ Log debugging information

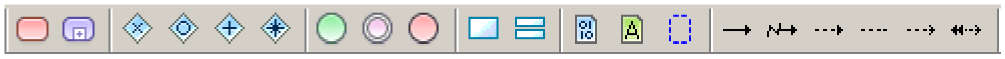
Fill the appropriate values and then press Apply button. The database server and database name must be the same as the ones specified in the Project settings dialog, in the Database tab.

Note: The security settings specified here could be different from the security settings specified in the Project settings dialog, in the Database tab. This is because ActiveFlow web application runs under a different account and with different privileges then the ActiveModeler Avantage. Also it is possible to use certain security credentials when creating and accessing the database from Avantage and other security credentials when ActiveFlow workflow system accesses the database.

The activeFlow workflow system logs the events, errors, debugging information usually in C:\Programs Files\ActiveFlow\Bin\Log. Please check the appropriate file if the ActiveFlow service does not start.

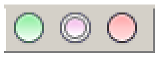
Diagram entities – modeling a workflow

Using ActiveModeler Avantage, the designer will define the business process. Described below are the entities used for designing a workflow. They can be selected from the ActiveModeler Avantage taskbar.



Events

Every workflow must have at least one start and one end events.

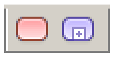


Regarding the start and end points there are the following restrictions:

- The workflows should have only one start point. If a workflow has 2 start points which merge in an AND-Join (illegal but tolerated map) the copy form functionality will have unpredictable results.
- It is possible to define a workflow with multiple end points but at run-time, there must be only one active.

Activities

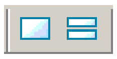
An activity denotes a task performed by a person (it is possible to define a robot-type activity but this will be described later). The activity can be placed in the map by selecting the activity button from the toolbox and then clicking on the map.



Each activity must have a caption and an ID. A unique ID will be automatically assigned by ActiveModeler.

Pools and lanes

For ActiveFlow the pools and lanes models the departments, sub-departments and the roles of the organization.



Links

A link connects 2 or more activities and transports data between them. Each link has a source activity and one target activity. An activity may have any number of incoming or outgoing links.

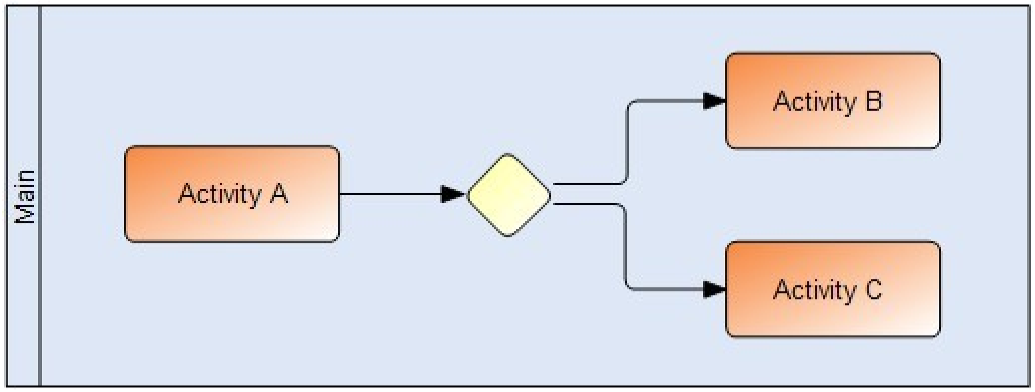
The activities can be linked by selecting the link button from the toolbox and dragging a line between the desired activities.

Gateways

Process maps are composed of groups of activities. There are two basic types of join and two basic types of split. With these four types of connection you can describe a workflow of any complexity.

OR-Split/OR-Join

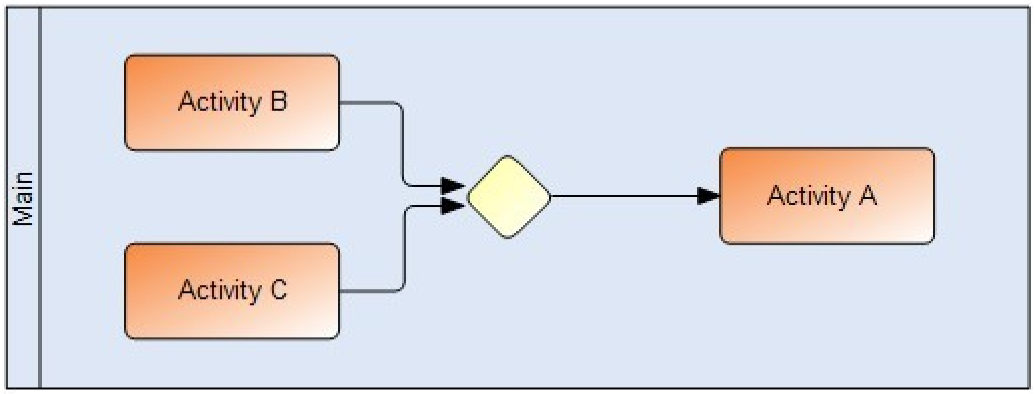
OR-Split



Use this kind of output connection to conditionally route a workflow depending upon form transaction processing responses, database lookups, and so on.

To do this you must add scripted logic in the **Transition condition** (for HTML forms) or **Pre condition** (for web forms) handler of Activity A's workflow rules property. The ActiveFlow Rules dialog will assist you in doing this.

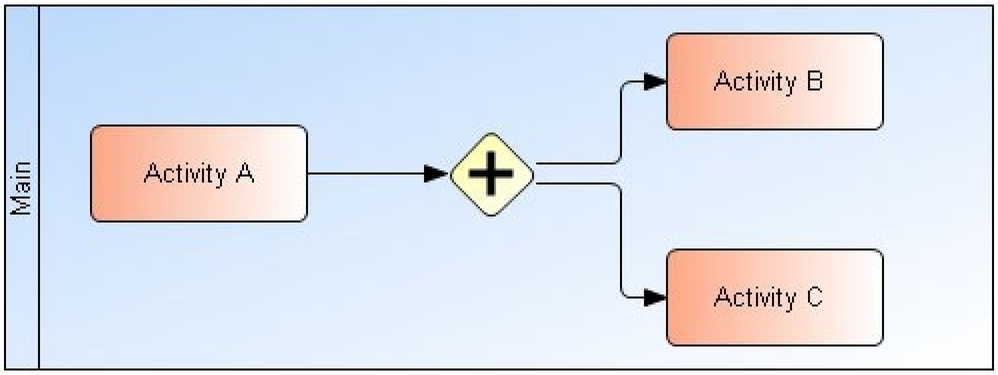
OR-Join



The simplest form of path combining is the simple OR-Join. In this type of input connection, input from any of the feeder links results in the target activity receiving an assignment. The OR-Join link must be paired by a previous OR-Split output connection.

AND-Split/AND-Join

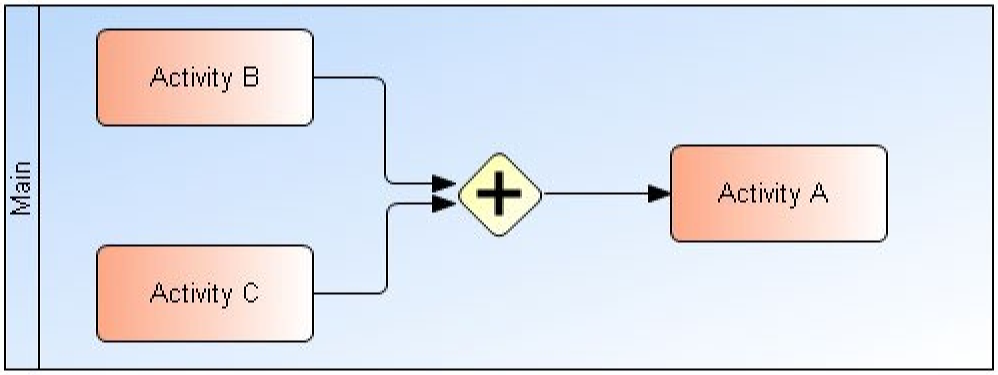
AND-Split



This type of workflow is very common. Here *Activity A* has an AND-Split output link and sends the output both to *Activity B* and *Activity C* .

AND-Split processes cannot be used for conditional path processing.

AND-Join



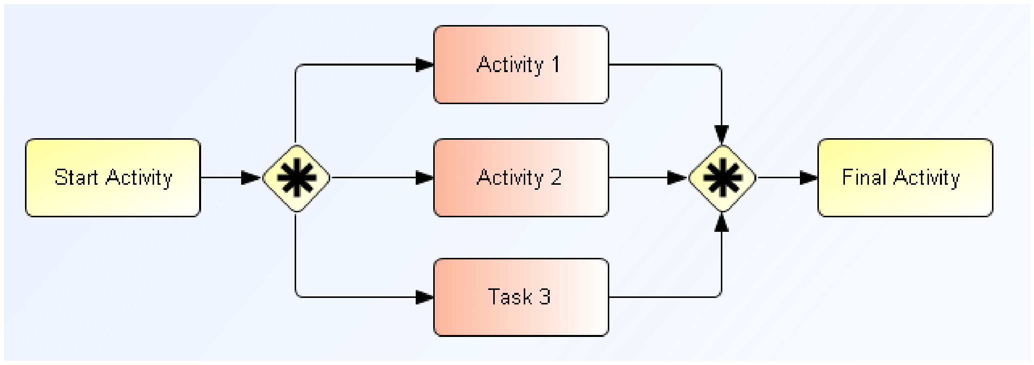
Sometimes an activity must be scheduled when a certain combination of inputs have arrived. This is called an AND-Join (or Waiting join). In this type of input connection, the target activity does not receive a work assignment until all feeder activities have been completed.

Complex-Split/ Complex-Join

There is another type of split and join supported by ActiveFlow. This is the "Complex type". This type of splitting enables more than 1 link to be followed whereas an OR-Split only allows one.
To set the run-time active links, the designer must write the proper code in the **Transition Condition** (for HTML forms) or **Pre condition** (for web forms) function. The ActiveFlow Rules dialog will assist the designer to implement this.

EXAMPLE

In the following example the split mode of the **Start activity** is **Complex-Split** and the join mode of the **Final activity** is **Complex-Join**.



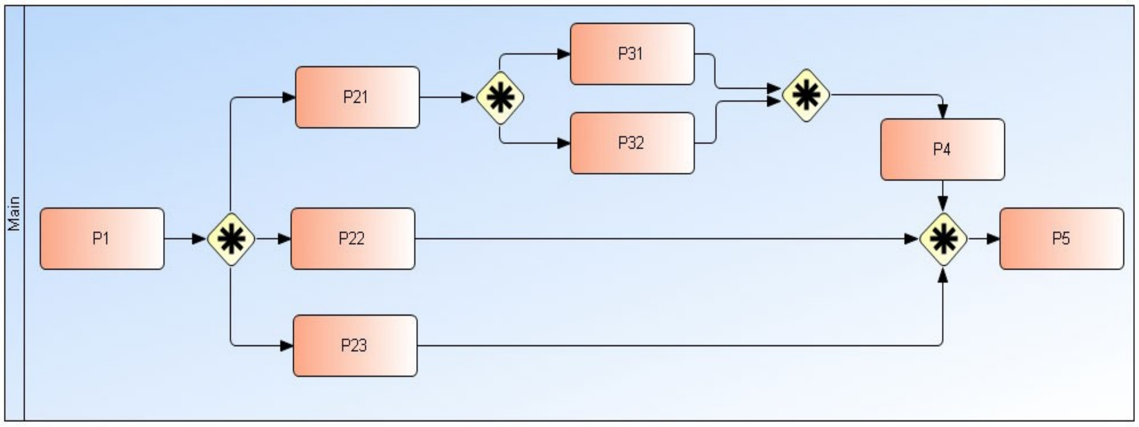
In the **Start activity** the active links at run-time are chosen according to the **Quantity** field on the attached form. The algorithm is: if the value is less than 10 then the active link will be L1, if the value is greater than 10 then the active links will be L2 and L3.

Limitations:
Despite the flexibility offered by this type of splitting, there are some limitations as follows:

- There can be no more Complex-Splits until the Complex-Join is resolved
- There must be the same number of links emerging from the Complex-Split as there are links merging into the Complex-Join
- Between a Complex-Split and a Complex-Join the map is not allowed to split
- Between a Complex-Split and a Complex-Join any type of split must be resolved and closed by the matching join.

Listed below are several **incorrect** process map examples:

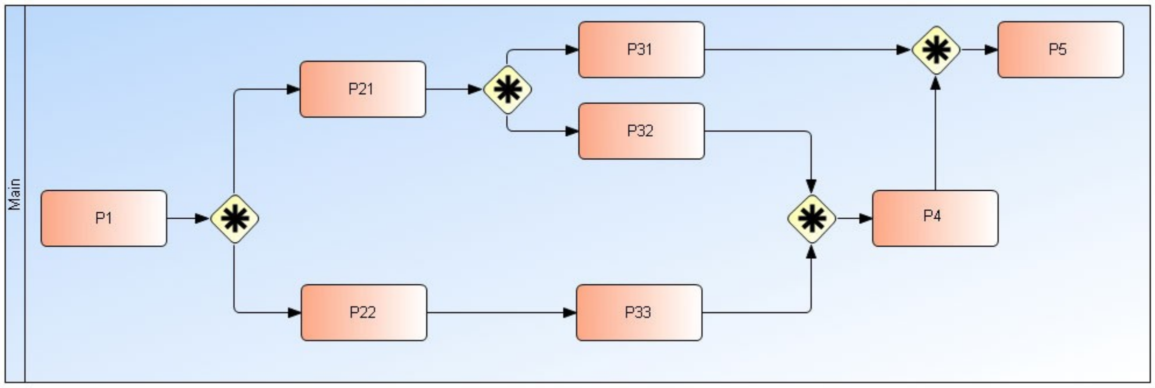
EXAMPLE



Assumptions
Complex-Splits: for P1 and P21
Complex-Joins: for P4 and P5

Mapping error
The Complex-Split in P1 is not closed before the Complex-Split in P21

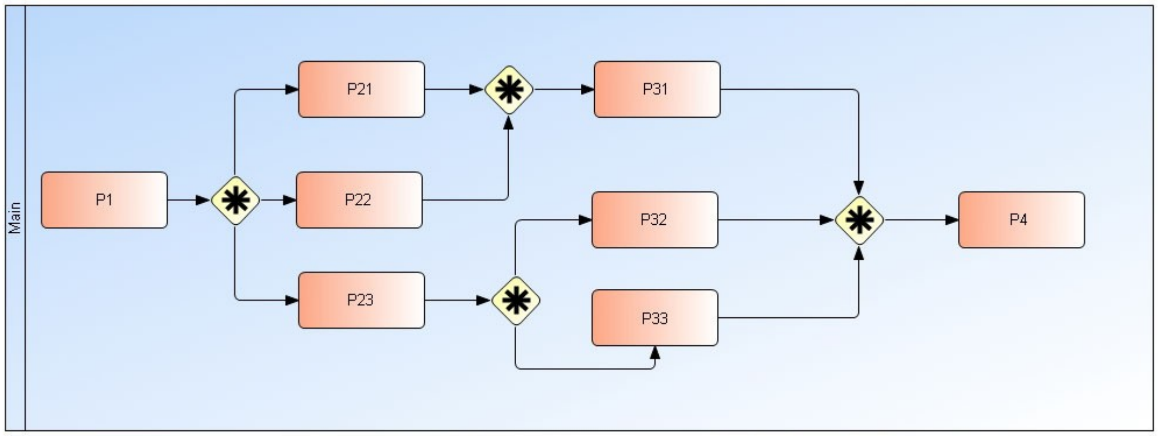
EXAMPLE



Assumptions
Complex-Splits: for P1
Complex-Joins: for P4
AND-Split: for P21
AND-Join: for P5

Mapping error
The process divergence in P21 (not all the messages merge into a Complex-Join)

EXAMPLE



Assumptions
Complex-Splits: for P1
Complex-Joins: for P4
AND-Split: for P23
AND-Join: for P31

Mapping error
The AND-Split in P23 is not closed before Complex-Join in P4, and there is no starting AND-Split for the AND-Join in P31.

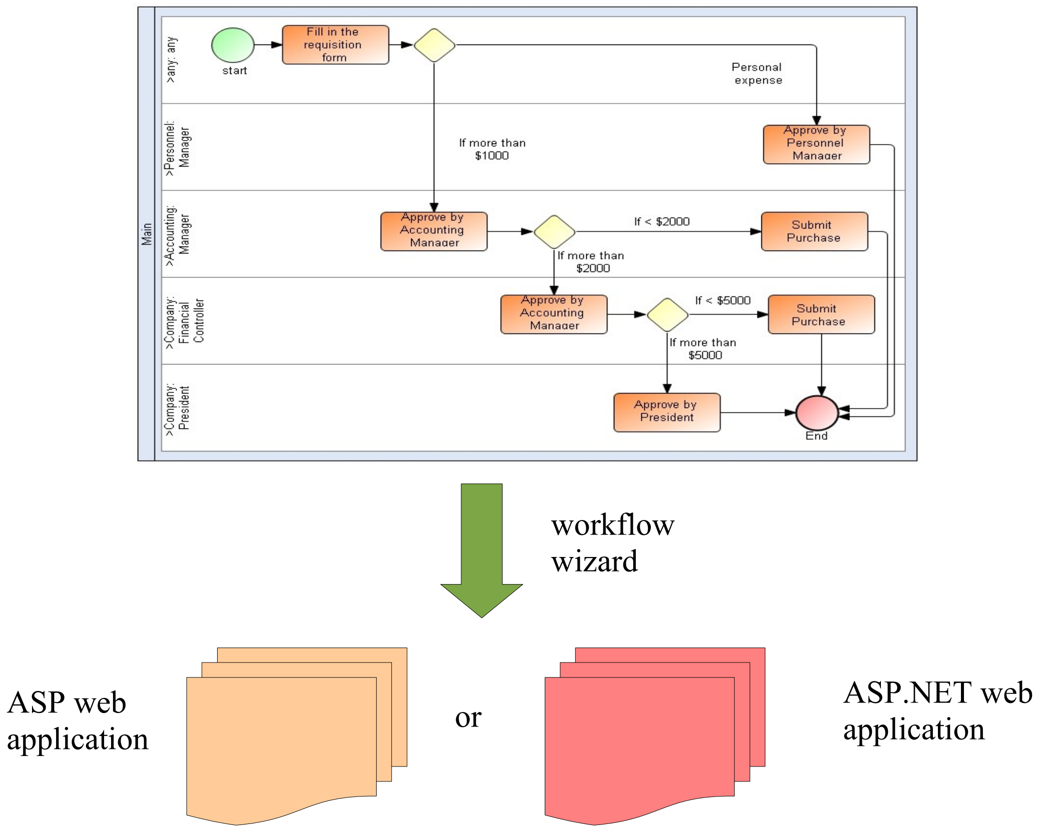
The example above is also incorrect if instead of the AND-Split and AND-Join there would be an OR-Split and OR-Join.

Workflow forms

ActiveFlow forms are end-user HTML documents used for entering and displaying workflow data and each activity in the workflow process should have a workflow form. The forms replace printed documents in a paper-based system.

For example, an ordering process map for a business might represent how an order is received and dispatched. The include form of such a process would be an order form, with fields for the customer name, order details, and so on.

As a workflow designer, you can choose between 2 types of forms to be used for handling and displaying data: simple HTML/ASP forms or web forms using .NET framework. For workflows with complex form functionality it might be more suitable to define web forms taking the advantage of the extensive functionality of the controls provided by the .NET framework whether for other more simple workflows, using just a static HTML form could be enough.



HTML/ASP forms and controls

For the HTML/ASP forms the workflow wizard copies the text from the form (between the <FORM> and </FORM> tags) and incorporates it into an Active Server Page file (.asp), which handles the transaction in the workflow system.

Follow these basic rules to create forms to be included in a workflow.

1. The form must have a **submit** button (with the HTML control type: **submit)**
2. Any scripts in your form must be defined inside the <Form> </Form> tags, otherwise they will not be included in the output ASP form being generated.
   Note: A good practice would be to define all the scripts (client or server) in external files which would be included in the form.
3. Use the ActiveFlow AF Controls for implementing the basic ActiveFlow functionality
   - give a title to the workflow
   - hold
   - attach file(s) to the form
   - return/reject
   - Cc
   - set expiry date
   - set form priority
4. Your form **must not** contain duplicate control names. This will cause a failure to generate a correct workflow. Usually, HTML editors automatically create unique control names, but some, create duplicates when you copy and paste.
5. Your form cannot have an **onsubmit** event handler. The Wizard installs an onsubmit handler in the client-side code. If you want to get control right before the form is submitted, you have to define a function called FormValidation() within the <FORM> tags.

FORM LOAD EVENT

When the form is loaded into the client browser, ActiveFlow performs the following operations:

- Initializes a set of JavaScript global client variables with the current user's properties. These global client variables are described below:

AF_UserID	Currently logged on user ID
AF_CrtFirstName	The **first name** of the currently logged on user.
AF_CrtLastName	The **last name** of the currently logged on user.

AF_CrtFullName	Full name of the currently logged on user.(<first name + last name> or <last name + first name> according to the ActiveFlow localization)
AF_CrtTitle	The **Title** attribute of the currently logged on user.
AF_CrtHierarchy	The **Hierarchy** value of the currently logged on user.
AF_CrtDepartment	The **list of departments** (separated by <,>) of the currently logged on user.
AF_CrtRole	The **list of roles** (separated by <,>) of the currently logged on user.
AF_LanguagePreference	The language preference of the currently logged on user.
AF_CrtGroupName	The **list of groups** (separated by <TAB>) where the current user belongs.
AF_MakerDeputiesList	The **list of users** (separated by <TAB>) for who the currently logged on user is a deputy maker.

- Writes the form values into the appropriate fields.
- Calls the function **SetDefaultValues** . Here is the place where you can implement workflow custom behaviour (e.g. enable/disable/show/hide form fields according to user's position within the company, set the fields' default values etc). To do so, define the JavaScript function **SetDefaultValues()** in the workflow form inside the <form></form> tags.
  A good practice is to define the actual logic outside the form in the include files as below:

```
<script type="text/javascript" src="Include/MyScripts.js"></script>
<script type="text/javascript">
function SetDefaultValues(){
  InitWorkflow_A();
}
</script>
```

It is possible to mix the client-side script with server-side script in order to implement the required behaviour. An example of such a combination could be used in the case where certain fields have to be read-only for the maker.

```
<script type="text/javascript">
var bIsMaker;
<%
nUserType = AF_GetCurrentUserType
if(nUserType = 0) then
%>
bIsMaker = True;
<%
else
%>
bIsMaker = False;
<%
end if
%>
</script>
```

You can define the above code in an external file and include it into the form:

<!--#INCLUDE FILE="<file_path>"-->

Doing this will allow you to re-use the code in other workflows as well.
When loading the form, the web browser will initialize the custom global variable **bIsMaker** with the appropriate value and you can call the function for making controls read-only in the InitWorkflow_A() function.

For the complete list of server-side functions please check the API chapter.

Form submit event

When the form is submitted ActiveFlow performs the following operations:

- Displays the submit confirmation message if the appropriate check-box was selected in the workflow wizard.
- Calls the function **FormValidation**. Here is the place where you can implement the custom validation of form fields (e.g. check mandatory values are input, check if values are within a given range, etc). To do so, define the JavaScript function **FormValidation()** in the workflow form inside the <form></form> tags.
  A good practice is to define the actual logic outside the form in the include files as below:

```
<script type="text/javascript" src="Include/MyScripts.js"></script>
<script type="text/javascript">
function FormValidation(){
  ValidateWorkflow_A();
}
</script>
```

- If the function **FormValidation** returns **True**, the ActiveFlow framework finalizes other internal operations and then sends data to the server, otherwise it will abort the submit.

Note:

- The form is submitted when the user presses the submit button in the form or when the user presses the ENTER key.
- All the global variables are available when the form is submitted and can be used in the custom form validation function.

FORM CONTROLS

The workflow software places certain restrictions on the names and IDs of HTML controls in the form. If you are designing forms to be used with the workflow system you must pay attention to these. The table below shows details.

HTML Control	Appearance	ID Value	Name	Comments
Simple Button		Any	*button*	Name must contain the string **button**
Submit button		Any	*submit*	Name must contain the string **submit**
Reset button		Any	any	
File upload control		Any	*_stdfile*	Name must contain the string **_stdfile**
Checkbox		Any	*checkbox*	Name must contain the string **checkbox**
Radiobutton		Any	*radio*	Name must contain the string **radio.** Radio buttons from the same group (mutually exclusive) must have the same name but each radio button must have a different value attribute.
Select list	Select Rules	Any	Any	
One-line edit box		Any	Any	
Scrolling edit box		Any	Any	

FORM FIELD SPECIFICATIONS

The restrictions on the control naming have been covered in the table above. Let's now go through other items you have to be aware of.

ActiveFlow does not use the ID of the field, but HTML specifications say that the ID of the control should be unique within a form. ActiveFlow requires that each control in the form must have a unique name, except radio button controls in the same group.

ActiveFlow uses some key words internally, so forms must not contain any control with the following names:

_AFFormTitle	_AFRRComments	_AFRejectReturn	_AFSubmitReturnP
_AFSubmitReturnM	_AFSubmitReject	_AFRR	_AFFileList
_ATTCHFILE<number>	_AFSendFwdAttch	_AFCCNameList	_AFApprove
_AFFiles	_AFCc	_AFSelCcUser	CcUserSelBtn
CleanBtn	SendCcBtn	_AFCCList	_AFCCComments
_AFSelUser<number>	_AFSelUserBtn<number>	_AFExpireDate	

Also, the form designer must not use the following keywords as function names for JavaScript or VBScript functions:

AF_UserID	AF_CrtRole	AF_CrtDepartment	AF_CrtHierarchy
AF_CrtFirstName	AF_CrtLastName	AF_CrtFullName	AF_CrtTitle
AF_MakerDeputiesList	AF_LanguagePreference	_AFSetAction	_AFOpenFile
_AFSetUser<number>	DisableSend	CcSel	confirmSubmit
SubmitHandler	CleanList	PreFlight	itsParameter
IsNumericData	ValidateAlert	ValidateControl	AFMessage
CheckRejectReturn	PreValidate		

Important:

The form design is very important to present a good user interface. Indeed a successful workflow implementation will depend on it. A good color scheme, self validating controls, etc will have a great impact on end users.

ActiveFlow HTML Controls (AF Controls)

The AF Controls are included in the form if you copy/paste the html code from the AF Control html files. These html files are located in:
Avantage\Plugins\ActiveFlowDesigner\ActiveFlow\StandardForms\AFControls

Form Title AF Control (control_Title.htm)

This AF Control enables the forms designer to assign a title to each form and to specify whether the title can be edited.

Hold form AF Control (control_Hold.htm)

This AF Control will insert a simple HTML button in the form with the **Hold** caption. The designer should use this AF Control in order to place a button in the form which allows the user to hold the form and place it in the hold list.

Hold button (temporary saves the form)

User AF Control (control_User.htm)

This AF Control will insert a simple HTML button in the form with **Browse...** caption.

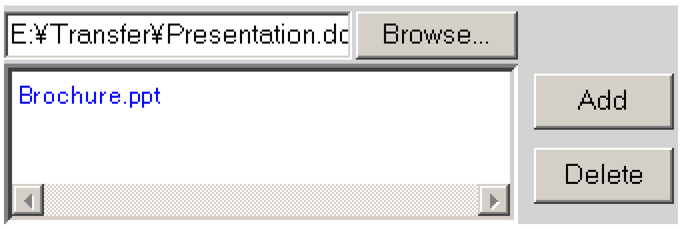
When you press the button a dialog box appears displaying the organization structure down to the user name.

File AttachmentEx AF Control (control_Attachments_full.htm)

The File AttachmentEx (Extended edition) has three features:

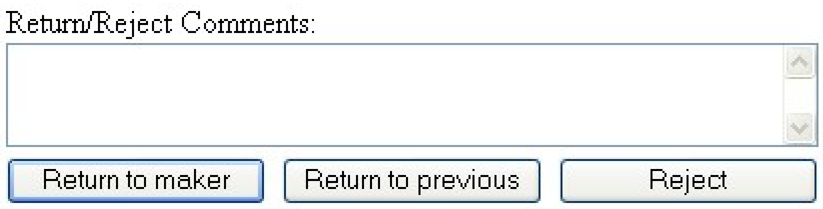
- Attach files to the form
- Delete files attached by the current user or by previous user(s)
- View/download files attached by the current user or by previous user(s)

The control is displayed below:



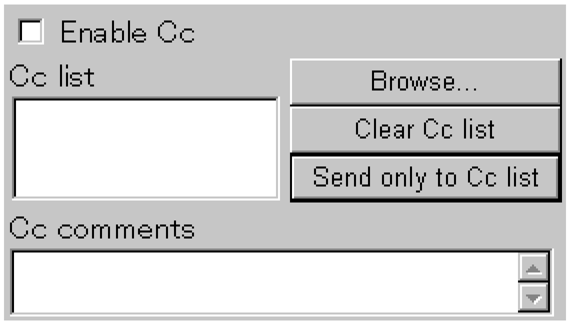
Reject/Return AF Control (control_Return_Reject.htm)

The Reject/Return AF Control adds buttons to a form, enabling the user to return a form back to the maker, to the previous user or to reject it.



Cc function AF Control (control_Cc.htm)

The Cc AF Control allows the designer to add a cc function to a form. Using this control, a copy of the form can be sent to all the selected users on the cc list as a means of passing information.
An approver can, before approving a form, use the **Send only to cc** list button to send the form to the users on the cc list. In this case, the user may re-open the form later for action.



Expiry date AF Control (control_ExpDate.htm)

This AF Control allows the designer to let the user specify the expiry date of the job sent to the next approver(s).

Job priority AF Control (control_Priority.htm)

By placing this AF Control in the form, the workflow designer allows priority setting .

Priority field (highlights the forms with high priority in the Intray)

Calendar AF Control (control_Date.htm)

By placing this AF Control in the form, the user can select a date using the standard ActiveFlow calendar control.
 In the form, the user will open the calendar control by clicking on the calendar image as shown below.

Date field (text field with a pop-up calendar):
- the name of the field "myDateField" can be replaced
- the line <script type="text/javascript" src="AFInclude/calendar.js"></SCRIPT> must be included only once in the page

Display the workflow status

ActiveFlow provides the framework to display the status of the workflow in the workflow form. You have 2 options for implenting this:

- display the workflow status as a list embedded in the form
- display the image of workflow map, each actioned activity highlighted

List embeded in the form - for this, include the standard ActiveFlow file FlowStatus.asp in your workflow form in the desired place.

<!--#INCLUDE FILE="FlowStatus.asp"-->

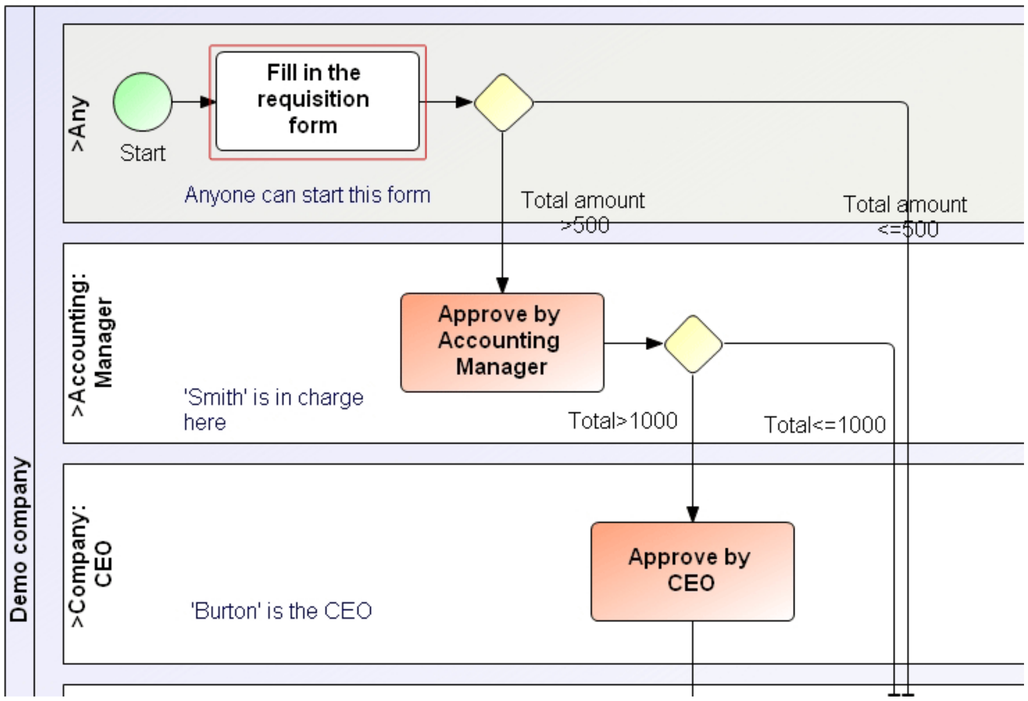
This include file will display the work item status in the form as below:

Activity name (Workflow name)	User	Date	User's comments	Status	Handled by

Image of the workflow map - for this include the standard ActiveFlow file ViewWorkflowStatus.asp in your workflow form in the desired place.

<!--#INCLUDE FILE="./HTMLMaps/ViewWorkflowState.asp"-->

This will insert a link in the form and by pressing it ActiveFlow will display the process map image with each actioned activity highlighted.



Web forms (ASPX) and controls

When creating the web forms (ASPX forms) the workflow designer should follow the following restrictions:

- The programming language must be C#
- When you create the aspx file, you must place the C# code in a separate file.

• The aspx file must contain the following page directive

<%@PageLanguage="C#"CodeFile="[filename].aspx.cs" Inherits="[className]" %>

• The aspx file must not contain the <%@ MasterType %> directive and the <asp:Content> tag

Also the workflow wizard will remove the following tags and replace them with new ones containing data relevant to ActiveFlow: <!DOCTYPE>, <html>, <head>, <title>, <body>, <form>.

A good start is to use the folllowing templates for both the aspx and cs files:

ASPX file

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="MyCode.aspx.cs"
Inherits="MyTemplate" Title="Template Page" EnableEventValidation="false" %>
<html>
   <!-- write the content here -->
   <form runat="server">
   ....
   </form>
</html>
```

CS file

```
using System;
using System.Collections;
using System.Configuration;
using System.Data;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.HtmlControls;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
public partial class MyCode : System.Web.UI.Page
{
...
}
```

Form load event

When the form is loaded into the client browser, ActiveFlow performs the following operations:

- First, on the server side initializes a global server-side variable whose properties are current user's properties.

FormContext.User.UserId	Currently logged on user ID
FormContext.User.CrtFirstName	The **first name** of the currently logged on user.
FormContext.User.CrtLastName	The **last name** of the currently logged on user.
FormContext.User.CrtFullName	Full name of the currently logged on user.(<first name + last name> or <last name + first name> according to the ActiveFlow localization)
FormContext.User.CrtTitle	The **Title** attribute of the currently logged on user.
FormContext.User.CrtHierarchy	The **Hierarchy** value of the currently logged on user.
FormContext.User.CrtDepartment	The **list of departments** (separated by <,>) of the currently logged on user.
FormContext.User.CrtDeptName	The list of departments names (separated by <,>) of the currently logged on user.
FormContext.User.CrtRole	The **list of roles** (separated by <,>) of the currently logged on user.
FormContext.User.CrtRoleName	The list of roles names (separated by <,>) of the currently logged on user.
FormContext.User.LanguagePreference	The language preference of the currently logged on user.
FormContext.User.CrtGroupName	The **list of groups** (separated by <TAB>) where the current user belongs.
FormContext.User.MakerDeputiesList	The **list of users** (separated by <TAB>) for who the currently logged on user is a deputy maker.

- Executes the functions OnLoadingData. This function can be used by the workflow designer in order to implement server-side custom behavior (e.g. enable/disable/show/hide form fields according to user's position within the company, set the fields' default values etc). To overwrite this functions, in the MyTemplate class define the functions OnLoadingData with the following signature:

```
protected override void OnLoadingData()
{
...
}
```

- After the form is loaded to the web browser, the ActiveFlow framework initializes a set of JavaScript global client variables with the current user's properties. These global variables are described below:

AF_UserID	Currently logged on user ID
AF_CrtFirstName	The **first name** of the currently logged on user.
AF_CrtLastName	The **last name** of the currently logged on user.
AF_CrtFullName	Full name of the currently logged on user.(<first name + last name> or <last name + first name> according to the ActiveFlow localization)
AF_CrtTitle	The **Title** attribute of the currently logged on user.
AF_CrtHierarchy	The **Hierarchy** value of the currently logged on user.
AF_CrtDepartment	The **list of departments** (separated by <,>) of the currently logged on user.
AF_CrtRole	The **list of roles** (separated by <,>) of the currently logged on user.
AF_LanguagePreference	The language preference of the currently logged on user.
AF_CrtGroupName	The **list of groups** (separated by <TAB>) where the current user belongs.
AF_MakerDeputiesList	The **list of users** (separated by <TAB>) for who the currently logged on user is a deputy maker.

- Writes the form values into the appropriate fields.
- Calls the client-side function **SetDefaultValues** . Here is the place where you can implement workflow custom behaviour (e.g. enable/disable/show/hide form fields according to user's position within the company, set the fields' default values etc). To do so, define the JavaScript function **SetDefaultValues()** in the workflow form inside the <form></form> tags.
  A good practice is to define the actual logic outside the form in the include files as below:

```
<script type="text/javascript" src="Include/MyScripts.js"></script>
<script type="text/javascript">
function SetDefaultValues(){
  InitWorkflow_A();
}
</script>
```

It is possible to mix the client-side script with server-side script in order to implement the required behaviour. An example of such a combination could be used in the case where certain fields have to be read-only for the maker.

```
<script type="text/javascript">
var bIsMaker;
<%
short nUserType = FormContext.API.GetCurrentUserType()
if (nUserType == 0)
%>
        bIsMaker = True;
<%
else
%>
        bIsMaker = False;
</script>
```

When loading the form, the web browser will initialize the custom global variable **bIsMaker** with the appropriate value and you can call the function for making controls read-only in the InitWorkflow_A() function.

For the complete list of server-side functions please check the API chapter.

Form submit event

When the form is submitted ActiveFlow performs the following operations:

- Displays the submit confirmation message if the appropriate check-box was selected in the workflow wizard.
- Calls the function **FormValidation**. Here is the place where you can implement the custom validation of form fields (e.g. check mandatory values are input, check if values are within a given range, etc). To do so, define the JavaScript function **FormValidation()** in the workflow form inside the <form></form> tags.
  A good practice is to define the actual logic outside the form in the include files as below:

```
<script type="text/javascript" src="Include/MyScripts.js"></script>
<script type="text/javascript">
function FormValidation(){
  ValidateWorkflow_A();
}
</script>
```

- If the function **FormValidation** returns **True**, the ActiveFlow framework finalizes other internal operations and then sends data to the server, otherwise it will abort the submit.

Note:

- The form is submitted when the user presses the submit button in the form or when the user presses the ENTER key.
- All the global variables are available when the form is submitted and can be used in the custom form validation function.

Form controls

For the web forms you can use the HTML controls for implementing the standard functionality such as form title, Cc, file attachment etc and the .NET controls for functionality which changed due to the .NET particularities.

- **Html controls** - are located in \AFNetControls\Html\ folder. To use them, the controls have to be included in the form using the <!--#INCLUDE FILE> directive.
- **NET controls**

1. History

To use the history .NET control you have to do the following steps:

– place the following code in the aspx file

```
<%@ Register TagPrefix="AF" TagName="history" Src="~/AFNetControls/Net/ history.ascx" %>
<AF:history ID="history" runat="server" />
```

– write the following code in the file containing the C# code for the web form

```
protected override void CustomCode_AfterOnLoad(){
        history.AF_API = AF_API;
}
```

- **ActiveFlow form handling buttons** - can be any html or web control button which calls the pre-defined _AFSetAction(n) function on the ***onclick*** event.
  The function parameter must have the following values according to the desired ActiveFlow action:

  0 - approve

  1 - return to maker

  2 - return to previous

  3 - reject

  4 - hold

  5 - CC only

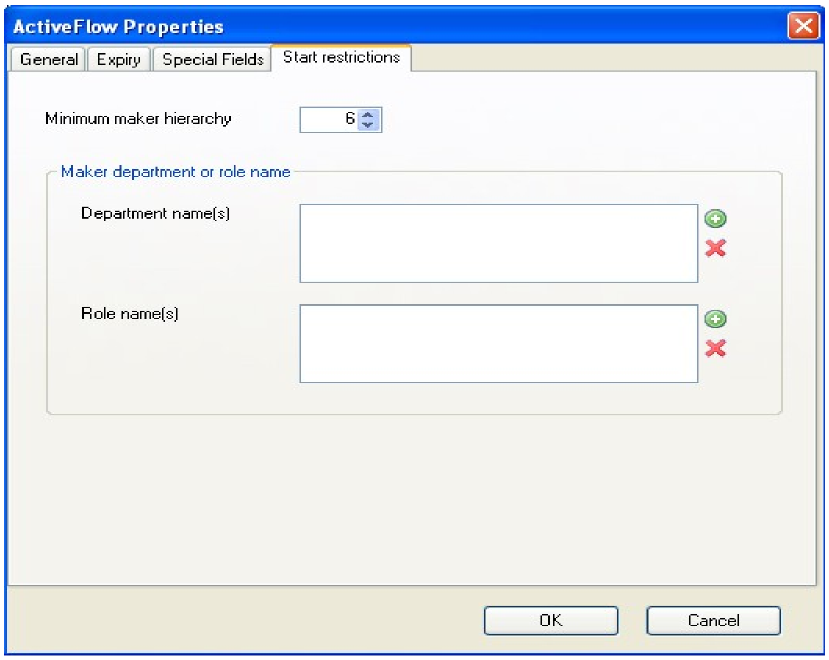
  6 - CC and submit

Workflow functionalities

Start restrictions

A user can start a new workflow if it is at least in one of the following situations:

1. The user is set as a candidate for the start activity or belongs to a group which was set as candidate for a start activity.
2. The user has a hierarchy level less than or equal to the **Minimum maker level** in the **Settings/Start restrictions** (see below) dialog associated with the start activity, and one of the following condition is met:
   - The user belongs to a department or role to which the start activity is linked to
   - The name of a department or role to which the user belongs to is set in the **Settings/Start restrictions** dialog;
   - If the user does not belong to an "**External**" department or role and the start activity is in one of the following situations:
     - it is not linked to any department or role;
     - it is linked to a department named "**Any**" ;
     - it is linked to a role named "**Any**" that is child of department named "**Any**" or a department to which the user belong to;


ActiveFlow Properties

General | Expiry | Special Fields | Start restrictions

Minimum maker hierarchy: 6

Maker department or role name

Department name(s):

Role name(s):

Notes:

The names "**Any**" and "**External** " are reserved for department and role names.
The **top level** in the organization is **0**.

Work assignment

In ActiveFlow work is assigned to users. There are several methods used by ActiveFlow for assigning work:

- direct assignament
- bubble-up routing
- group broadcast
- workflow pool

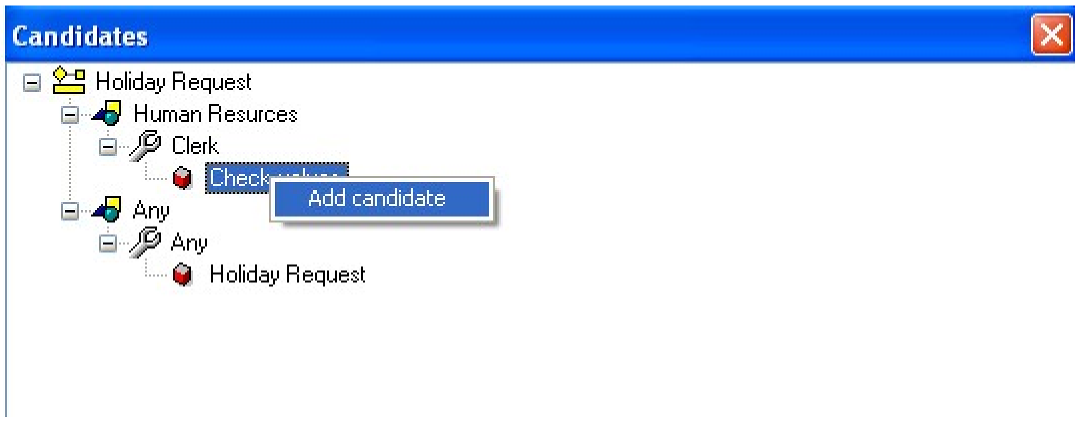
DIRECT ASSIGNMENT

For each activity in the process map, the potential target candidates for receiving work items can be set. Based on the number of jobs in the candidates' in-tray the ActiveFlow engine **will choose at run time** the actual person to whom the job will be assigned.
In order to assign candidates to the activities the designer will use the candidates control from ActiveModeler.

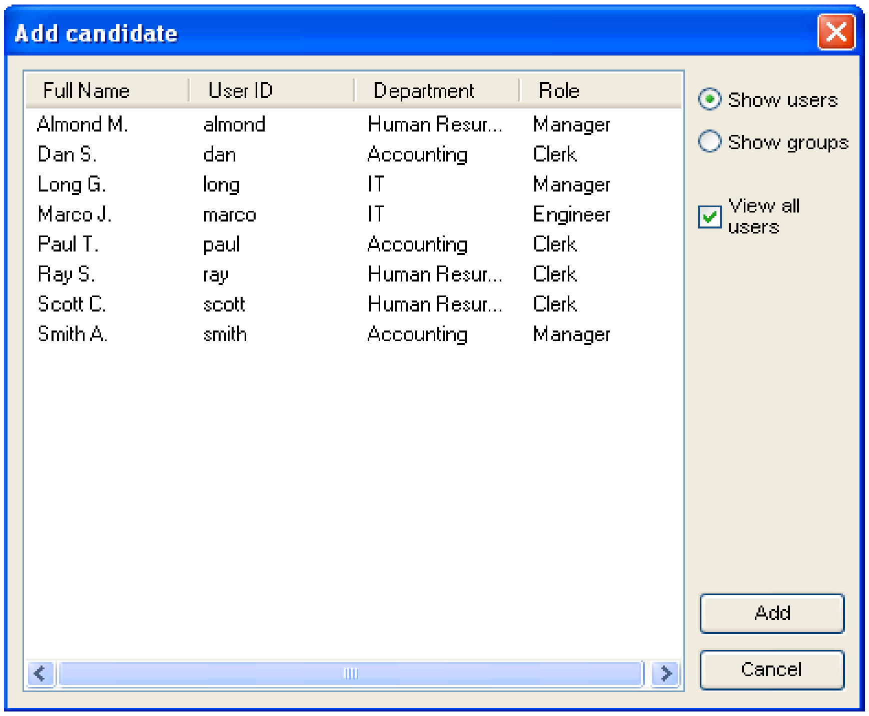
Important: If no user is assigned to the target activity and it is not used any other work assignment the ActiveFlow engine will choose a user (the one with the least number of waiting forms in his/her intray) from the department/role linked to the lane where the target activity is placed. If the activity is not under any role or department the form will be sent to the approver's supervisor.

For manually assigning a candidate to the activities, in ActiveModeler Avantage right-click the diagram item then select **ActiveFlow -> Candidates...**

The organizational structure and activities relating to the map appear. Then right-click the activity in the tree structure as below:



You can attach any number of users as candidates for an activity. Also it is possible to specify a group of users and in this case the ActiveFlow engine would choose one user from that group.



Full Name	User ID	Department	Role
Almond M.	almond	Human Resur...	Manager
Dan S.	dan	Accounting	Clerk
Long G.	long	IT	Manager
Marco J.	marco	IT	Engineer
Paul T.	paul	Accounting	Clerk
Ray S.	ray	Human Resur...	Clerk
Scott C.	scott	Human Resur...	Clerk
Smith A.	smith	Accounting	Manager

Show users [✓]
View all users [✓]

Although initially the control will display only the users under the department/role where the activity belongs, it is possible to view all the users from the organization and select any of them as candidate.

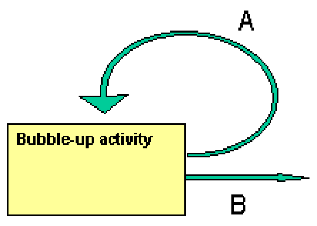
Notes:

1. It is important to note that for this kind of routing, the work will be assigned **to only one** user.
2. This kind of work assignment is available only if the activity does not have set any of the following properties: bubble-up routing, group routing or pool.

BUBBLE-UP ROUTING

Often is necessary for a form to get the departamental approval before further processing. In this situation, at run-time the number of approval steps is variable depending on the user who is raising the request and the maker's position within the department.

Bubble-up routing means that starting from the maker, the authorization path will follow the chain of hierarchy automatically within the department. Logically the bubble up routing can be represented as below:



Each user has 2 attributes:

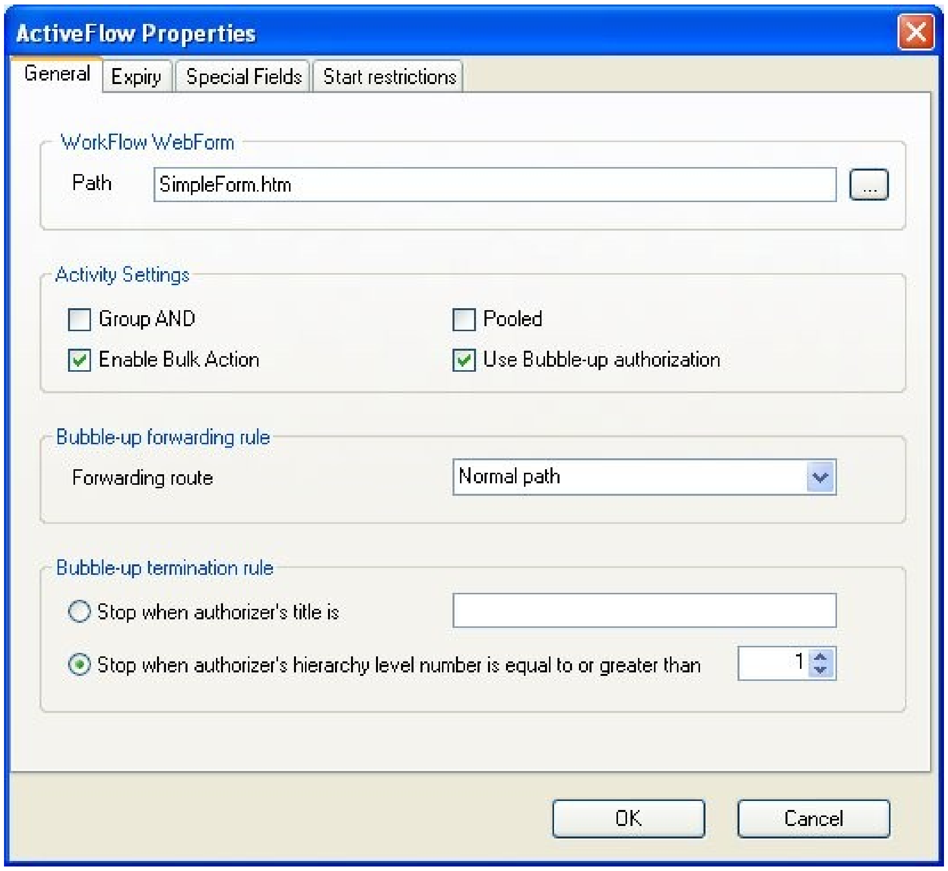
- A normal approval route
- An alternative approval route

which defines the user who will receive the form in the case of bubble-up routing. ActiveFlow provides 2 methods (or breakpoints) for stopping the bubble-up loop:

- When the approver has a certain title *or*
- When the approver has a certain hierarchy level

These items are also attributes of the user.

In ActiveModeler Avantage, the bubble-up routing is a property of the activity. To set it, right-click on the activity and choose ActiveFlow/Settings option. The following dialog is displayed.


ActiveFlow Properties

General | Expiry | Special Fields | Start restrictions

WorkFlow WebForm
Path: SimpleForm.htm

Activity Settings
- [ ] Group AND
- [ ] Pooled
- [✓] Enable Bulk Action
- [✓] Use Bubble-up authorization

Bubble-up forwarding rule
Forwarding route: Normal path

Bubble-up termination rule
- ○ Stop when authorizer's title is
- ● Stop when authorizer's hierarchy level number is equal to or greater than: 1

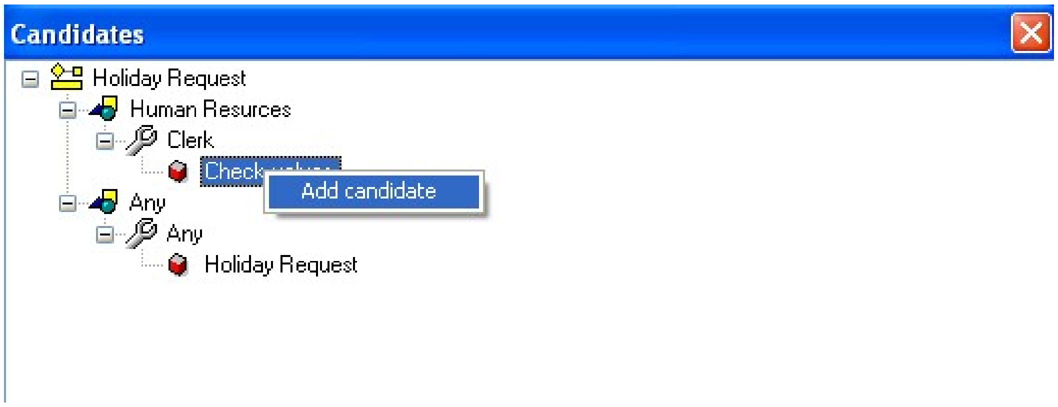
The workflow designer uses this dialog to select the stop condition rule and the bubble-up routing type.

Important!
Either the administrator or user is responsible for setting the normal or alternative path values. These values will be used in **all workflows** that use the bubble-up routing method.

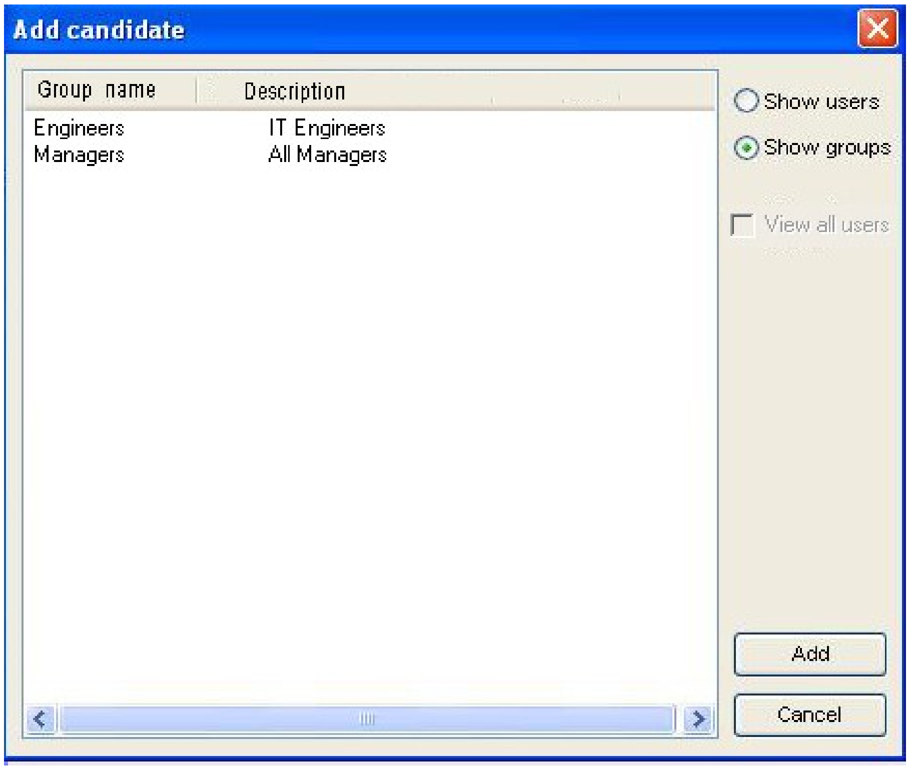
Group broadcast

For this type of routing the form will be sent to all the members of a group. To select this type of routing, you need to enable the relevant activity for group behaviour. To do this, right-click the activity and choose **Settings...** from the **ActiveFlow** menu . Then select the **Group broadcast** check box.

Also, the candidate must be a group in the Candidates control. To do this, choose the **Candidates...** option from the BPtree menu(right-click on diagram and select ActiveFlow). The organizational structure and activities relating to the map appear. Then right-click the activity in the tree structure as below:

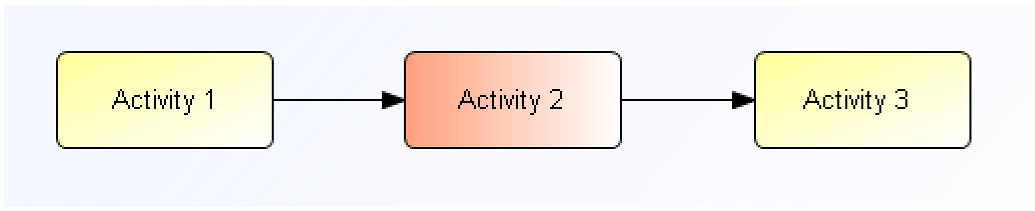


Then press **Add candidate**, choose the **Show groups** radio button, select a group, then **Add** the selected group for participation in the activity.



Group name	Description
Engineers	IT Engineers
Managers	All Managers

An example of a process map with group participation is shown below:



In this case, **Activity 2** has the **Group broadcast** check box set and has as a candidate all the members of group **G1**. In this case, the form from **Activity 1** will be sent to all the active members of group **G1**. The last activity in the chain (**Activity 3**) will have a status of **Waiting for approval** only when all the members of the group have sent the form forward. This equates to an AND-Join from all the users in group **G1** to the last activity: **Activity 3**.

Limitations
This type of routing has the following restrictions:

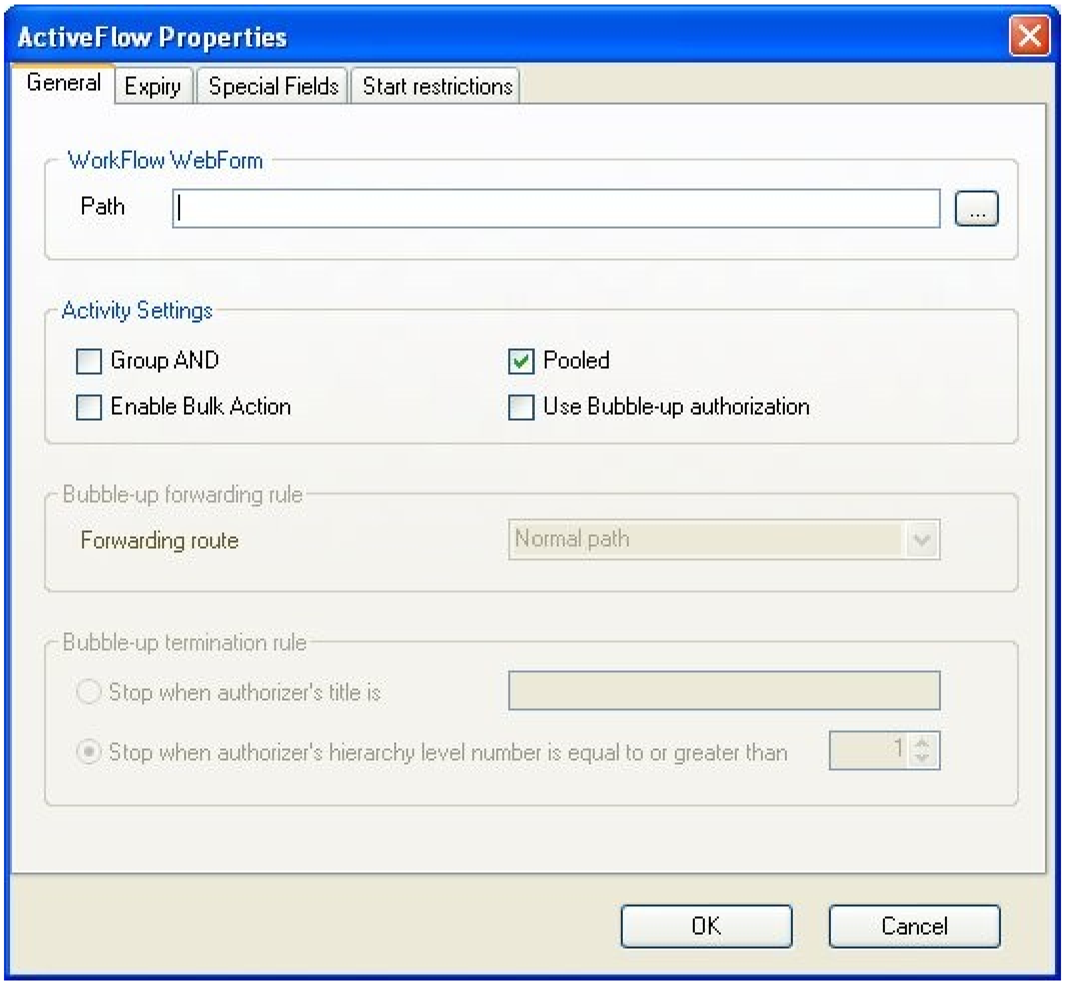
- Activity 2 must have only one input link and only one output link
- Activity 3 must have only one input link and the **group broadcasting** checkbox must not be selected.

Three such activities compose a unit for the Group broadcast feature, and this complete unit must appear in a process map.

WORKFLOW POOL

This functionality allows a user to send a form to a "pool" of work items.This pool will be associated to an activity in the map and any user who belongs to the same department/role as the activity may pick an item from the pool. If the pooled activity is in the department and role called "**Any**" then any user may pick the item from that pool.
Note: "Any" is a keyword.

In order to set the pooled attribute of an activity, from ActiveModeler open the **Settings...** dialog box of the activity and check the appropriate checkbox.


ActiveFlow Properties

General | Expiry | Special Fields | Start restrictions

WorkFlow WebForm

Path: |

Activity Settings

- [ ] Group AND
- [✓] Pooled
- [ ] Enable Bulk Action
- [ ] Use Bubble-up authorization

Bubble-up forwarding rule

Forwarding route: Normal path

Bubble-up termination rule

- ○ Stop when authorizer's title is
- ● Stop when authorizer's hierarchy level number is equal to or greater than 1

OK | Cancel

ActiveFlow rules wizard

Quite often in the real-world workflows the form has to be routed to one path or another according to a field value or other criteria and for such situations the workflow designer will have to use OR-Split or Complex-Split gateways. Also most of the times the workflows are not just stand-alone applications but they have to be integrated to various back-end systems or databases and the system must perform certain actions when a task is completed.

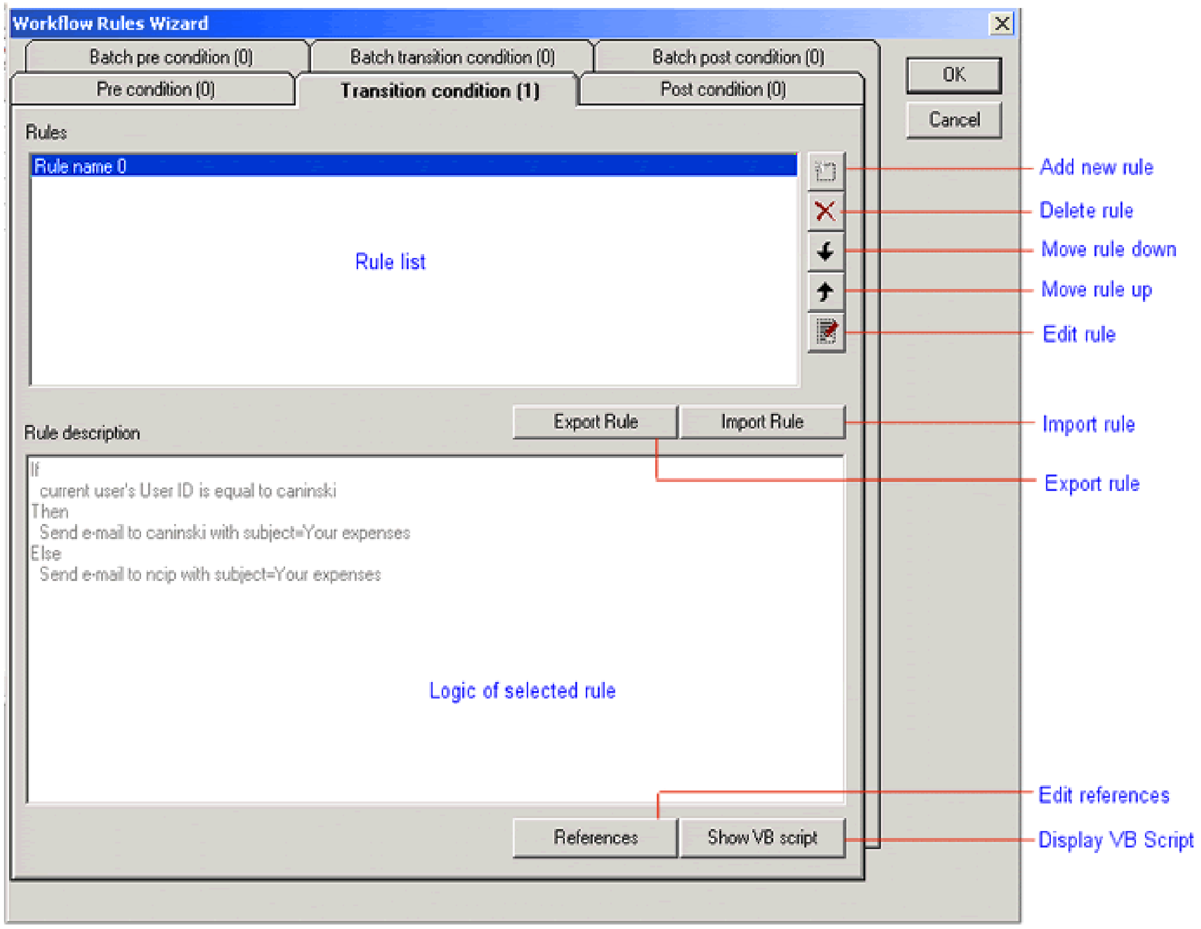
ActiveFlow provides a mechanism to implementing all this functions and you can access it by right-click on an activity shape in a process map and choose the **Rules...** from the **Workflow** menu.

Depending on the workflow type (ASP or ASP.NET), a different Rules Wizard dialog is presented. The differences between the 2 dialogs are as follows:

- The ASP dialog allows 3 types of conditions, namely Pre, Transition and Post conditions whereas the dialog for ASP.NET workflow allows only 2 conditions of Pre condition and Post condition. We believe it is sufficient for the workflow designer to get control only before and after ActiveFlow processes the transaction.

- The workflow wizard dialog for ASP.NET workflows combines batch action rules with the normal action rules because the APIs which can be used by the workflow designer are the same for the 2 types of actions. For ASP type workflows the APIs were different for normal action and batch action.
- The custom code written and also generated by the rules wizard for ASP workflows is VBScript whereas the code written in the rules wizard dialog for ASP.NET workflows must be C#

RULES WIZARD DIALOG FOR ASP WORKFLOWS



```
If
  current user's User ID is equal to caninski
Then
  Send e-mail to caninski with subject=Your expenses
Else
  Send e-mail to ncip with subject=Your expenses
```

As you can see, there are six tabs within the dialog. Each tab represents one of the six standard handlers that can be deployed for an ActiveFlow form transaction. The primary handlers are "Pre condition", "Transition condition" and "Post condition", these broadly correspond to the WfMC phases of a workflow transaction. The other three handlers are for bulk approval processing. Batch approval is a powerful feature of ActiveFlow whereby you can select a group of transactions and then approve them at once (this is very useful for example in an Accounts Dept. where there could be thousands of expense vouchers to approve which have already been approved by the Departmental Manager of the employee concerned - a high level check only is normally required). The same three phases of "Pre", "Transition" and "Post" provide the same phasing as for an interactive approval transaction.

Rules wizard controls

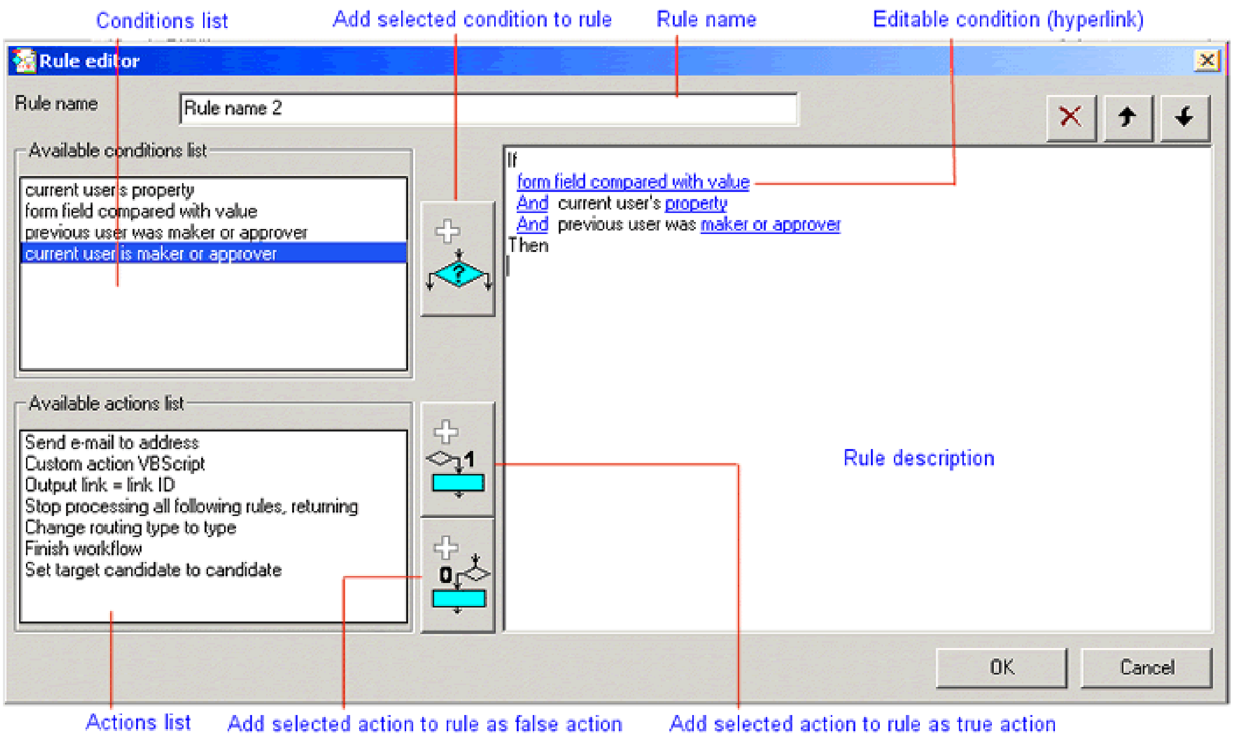
The rules list
The rules list shows a list of all the rules defined for the currently selected tab (handler). The list is in the form of "rule 0", "rule 1", "rule 2" etc. In order to apply an action to a rule (for example delete it, modify it or move it) you must first select it with the mouse. The caption of the tabs displays in brackets the number of rules associated with that handler.
The rule description
The rule description list shows an expansion of the steps contained within each rule. The steps contain the discrete actions that are performed within the rule. If you want to examine or modify the step double-click on the rule or click on the Edit rule button.

The New Rule button displays the Rule editor dialog. The rule editor dialog allows you to define the logic and steps that will be contained in the rule.
The Delete rule button deletes the currently selected rule.
The Move rule up/down buttons moves the selected rule up/down in the rules list. This allows you to change the rules ordering.
The Edit rule button displays the Rule editor window. This window lets you specify the conditions and actions for the rule. The rule editor window can be called by double-clicking on the rule.
The Import Rule buttondisplays the Open file dialog for choosing a .rule files. The rules stored in that file will be imported in to the current tab. Rules can be exported in a file with extension .rule by clicking **Export** button.
The Export Rule button displays the Save file dialog for specifying a path and a name for the .rule file. The selected rules are exported in a XML format to the specified file.
The References button displays a window used to specify files included into the workflow .asp files. This option is detailed at the end of this section.
The Show VBScript button displays a window that contains the generated server-side script code made so far for the handler. Please note that you can't change the code in this window, it is for viewing only. If you want to add "hand crafted" script then you should add a "custom" script to the rule with the appropriate button in the Rule editor dialog.
The OK button dismisses the Rules Wizard and updates the activity in the process model.
The Cancel button dismisses the Rules Wizard and discards any changes that you may have made to any of the rules of the activity.

Rule editor dialog

The Rule Editor Dialog allows you to define and edit the conditions and actions that are contained with a rule. In order to display the Rule Editor Dialog you must select a rule within the Rules List and then either click on the either **Edit rule** or **New rule** buttons.

This rule editor dialog contains lists of available conditions and available actions. You can build a new rule by selecting a condition or an action and pressing "Add Condition", "Add Action"


Conditions list
Add selected condition to rule
Rule name
Editable condition (hyperlink)
Rule editor
Rule name
Available conditions list
Available actions list
Rule description
OK
Cancel
Actions list
Add selected action to rule as false action
Add selected action to rule as true action

Rule name: Rule name 2

Available conditions list:
current user's property
form field compared with value
previous user was maker or approver
current user is maker or approver

Available actions list:
Send e-mail to address
Custom action VBScript
Output link = link ID
Stop processing all following rules, returning
Change routing type to type
Finish workflow
Set target candidate to candidate

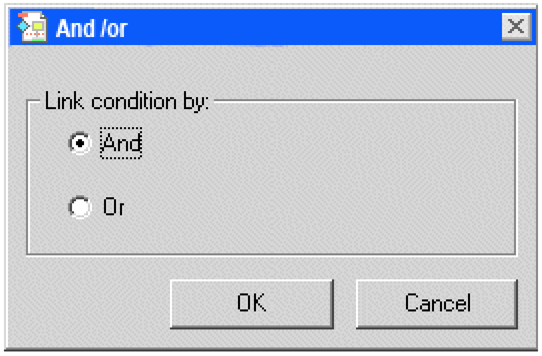
If
form field compared with value
And current user's property
And previous user was maker or approver
Then

or "Add Action Else" buttons. The selected actions or conditions can be edited by clicking the hyperlinks from the rule description window. The selected actions or conditions can be also deleted, moved up or down.

For example, we added all types of conditions and the generated statement is:

If <u>field compared to value</u>
<u>And</u> If current user's property is <u>property</u>
<u>And</u> If sender was <u>maker or approver</u>
<u>And</u> If current user is <u>maker or approver</u>

The parts of the text that look like hyperlinks are just that. Clicking on an underlined word or phrase causes the wizard to display a dialog that lets you change or review something. For example if you click on <u>And</u> you will see this And/Or dialog



Link condition by:
[✓ And]
Or

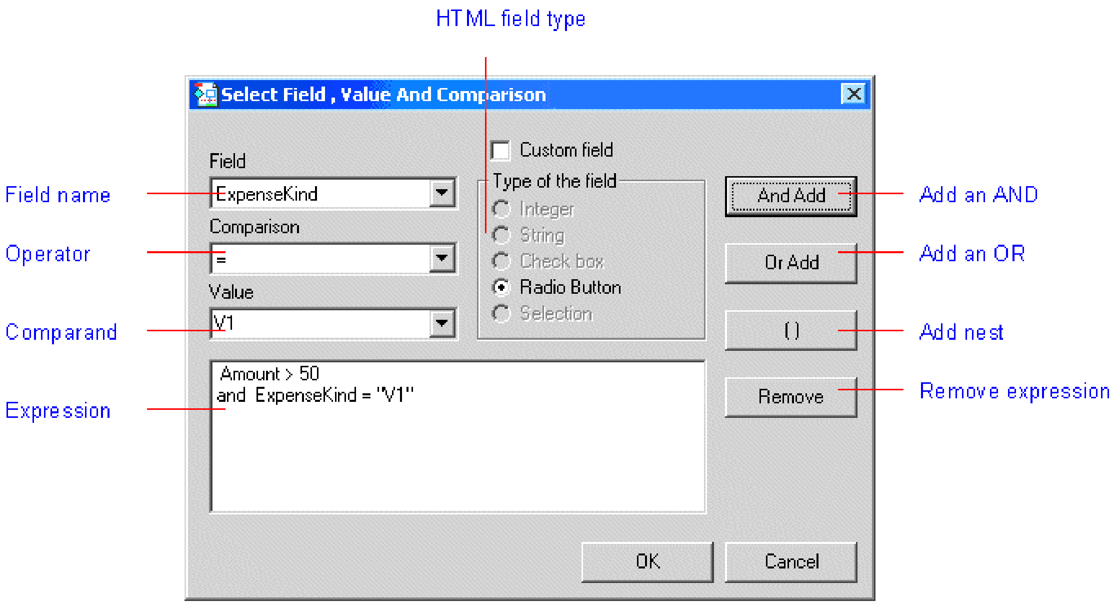
After changing the selection to OR the output in the expression window will now look like this:

If field compared to value
Or If current user's property is property
And If sender was maker or approver
And If current user is maker or approver

You have just changed the combination rule from AND to OR.

Rule conditions

1. If form field compared to value
This kind of rule lets you test the value of a named HTML form field. Depending upon the result of the comparison you can conditionally execute a workflow action. There must be an associated html form attached to the activity in order for this rule to work. (refer to Associated Documentation in the ActiveModeler Users Guide).



Field: ExpenseKind
Comparison: =
Value: V1
Type of the field: Integer String Check box [✓ Radio Button] Selection

```
Amount > 50
and ExpenseKind = "V1"
```

The aim of the Field comparison dialog is to allow you to produce compound conditional expressions based on the values of named HTML fields. For example:

If LastName = "Caninski" and Amount > 50
Then
Output link = L11850571 (NO)
and Send e-mail to User ID:caninski@dogmail.com Subject:Your expense claim Field

This is a drop down list that shows all of the field names that are defined in the HTML form which is attached to the ActiveModeler activity.
The field type allows you to specify the type of data the field contains. For example, an edit box is ambiguous because it can be interpreted as number or a string. The field type lets you overcome this ambiguity.

- Integer - The field can contain a whole number
- String - The field is interpreted to be a string. **Note**, a textarea box may contain multiple lines. These are delimited by vbCrLf.

- Checkbox - The field is a Boolean value
- Radiobutton - The field contains the value from a group of related radio buttons. The value field identifies the radio button
- Drop down list - The field contains the selection from a drop down list (combo-box) type control.

The comparison field contains a drop down list of comparison operators.

=	"Equal to". The test is True if the value is equal to the comparand.
<>	"Not equal to". The test is true if the value is not equal to the comparand.
>=	"Greater than or equal to". The test is True if the value is greater than or equal to the comparand
<	"Less than". The test is True if the value is less than the comparand.
<=	"Less than or equal to". The test is true if the value is less than or equal to the comparand.

ActiveFlow can use fields that are not visible from within a designer form. These fields are created dynamically either by the client code either by the asp code running in the web server. For example, the following client side code creates a series of custom fields named Value1, Value2, Value2 etc.

```
<script type="text/javascript" >
for(i = 0; i <10; i++){
  document.write("<br>New value <input name='Value'+i>");
}
</script>
```

The value field is compared with the data that is contained in the HTML form field value at run time.
The Remove button deletes the last line of the composed expression in the expression field list.
The Or ADD button adds a test with a logical OR test on the field/value expression.

If Amount > 50 or ExpenseKind = "V2" or LastName = "Smith"
Send e-mail to address

The And ADD button adds another test with a logical AND on the field/value expression.

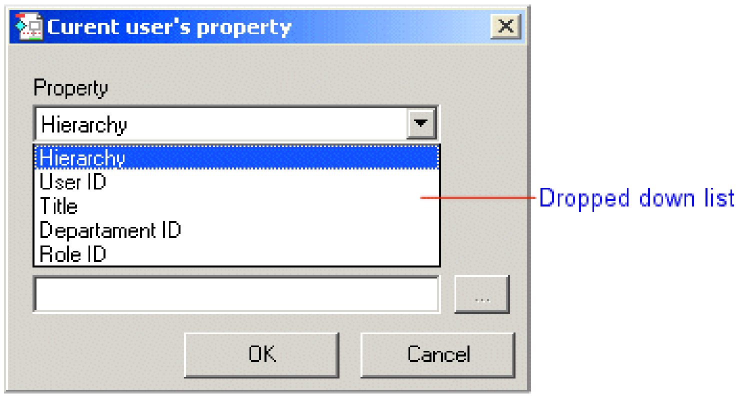
If Amount > 50 and ExpenseKind = "V1" and LastName = "Smith"
Send e-mail to address

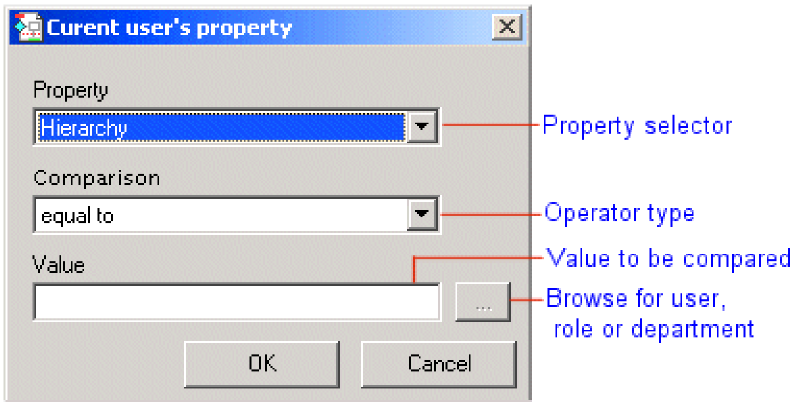
The ( ) Nesting button parenthetically nests an expression. By using this in combination with IF OR AND you can construct sophisticated conditional logic.

If (LastName = "Caninski" and Amount >50) or ExpenseKind = "V3"
Send e-mail to address

2. If current user property

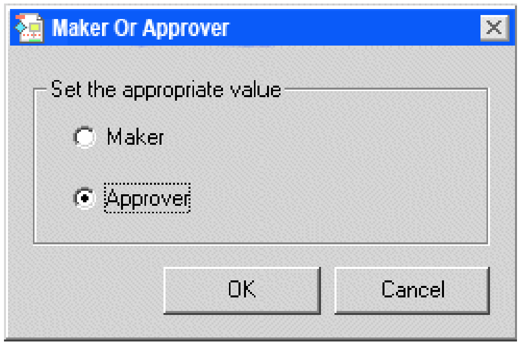
This kind of rule let you examine the public properties of the current user and compare it with a literal value. Some items in the ASP session object are automatically set by the workflow engine to reflect the properties of the currently logged in user. This information can be accessed from transition handlers. So, when a person opens work from their in-tray, the form will contain some useful information about the current user (for example their ID, their hierarchy, role, department etc). For more information see **ActiveFlow API Guide [UserObject].** The **If current user property is** test allows your form to make decisions based on these properties.





DepartmentID	A comparison is made between the department the user belongs to and the department ID entered into the value field.
UserID	A comparison is made between the user ID and the user ID entered into the value field.
Hierarchy	A comparison is made between the hierarchy level of the user and the hierarchy number entered into the value field.
Title	A comparison is made between the title of the current user and the title entered into the value field
RoleID	A comparison is made between the role of the user belongs to and the role ID entered into the value field.

3. If previous user was maker or approver
This rule lets you conditionally perform an action depending upon whether the previous user (i.e. the sender) of the form was a maker or an approver. ActiveFlow classifies users into two categories : Makers and Approvers. A maker is someone who fills in a blank form (i.e. an employee who fills in an application for a pay rise is a maker). An approver is any person or agent who subsequently must check the information entered by a maker.



Maker [✓ Approver]

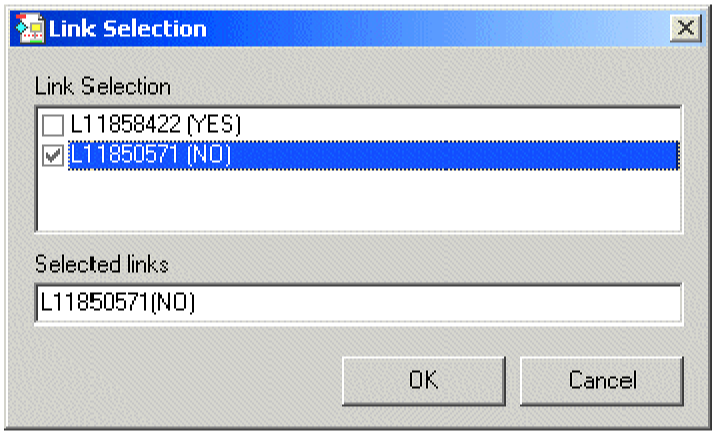
4. If current user is maker or approver
This rule lets you conditionally perform an action depending upon whether the current user is a maker or an approver. This test is the same as the " **If previous user was maker or approver** " test with the exception that it refers to the current user.

Rule actions

You can add actions to you rule by selecting an action in the available actions list and pressing the
Add Actionor the **Add action else.** Add action button adds an action on the true branch of the condition. That is the actions added with the Add action button are executed when the condition or conditions added to the rule are met. The actions added with the Add action else button are added on the false branch of the condition, that is they are executed when the condition(s) specified are not met.
In case there is no condition added to the rule you can add actions only with Add action button. These actions are executed always, that is every time the handler that contains the rule is executed.
Here is a description of the available actions:

1. Link = Link ID

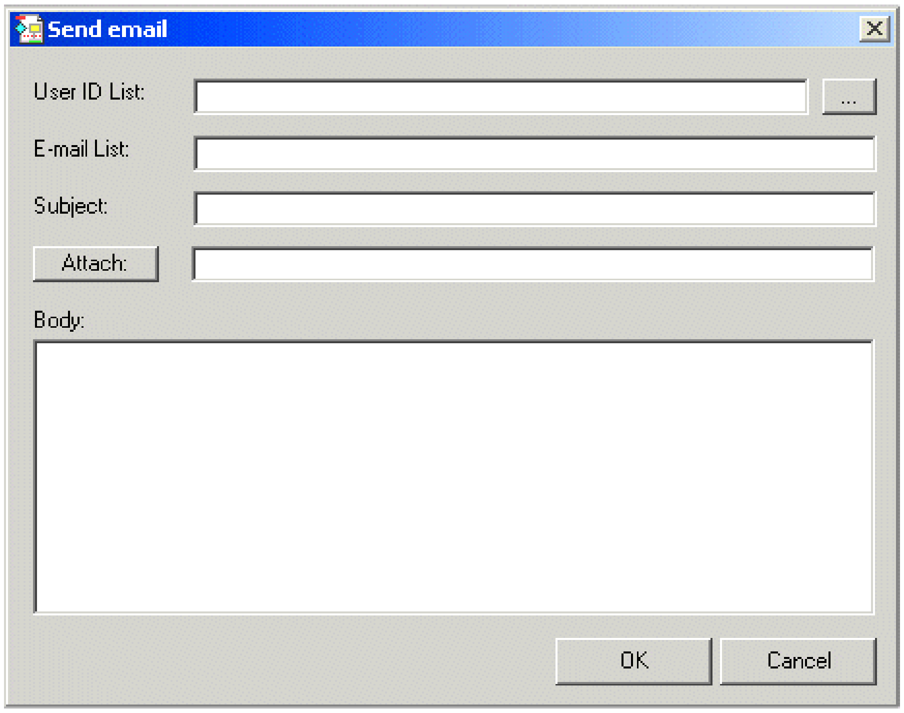
This action controls the direction of the transaction. By selecting a link you steer the transaction to the activity in the ActiveModeler map that is connected by that link. The rules wizard works with Link Ids. These are strings automatically assigned by ActiveModeler in the form of GUID ({9FBD652E-C2E5-4502-B551-EF06FFEEDFAF}). These strings are unique within a process model and they are automatically changed if you Save as or copy and paste linked activities. You can change a link ID within ActiveModeler by selecting the link and then choosing ID from the Properties toolbar. The Link selection dialog also shows the link caption within parentheses so you can tell at a glance which link you are referring to. In the example below, the link captions are "YES" and "NO".



If you make multiple selections the workflow engine will send the form to each link in the selection.

2. Send e-mail to address

This action will result in an email message being sent to the specified email addresses. You can specify subject matter, cc list and body text and an attachment. A typical example would be to advise interested parties if a certain condition occurred during a transaction (for example if a user has entered a spurious or doubtful expense request).

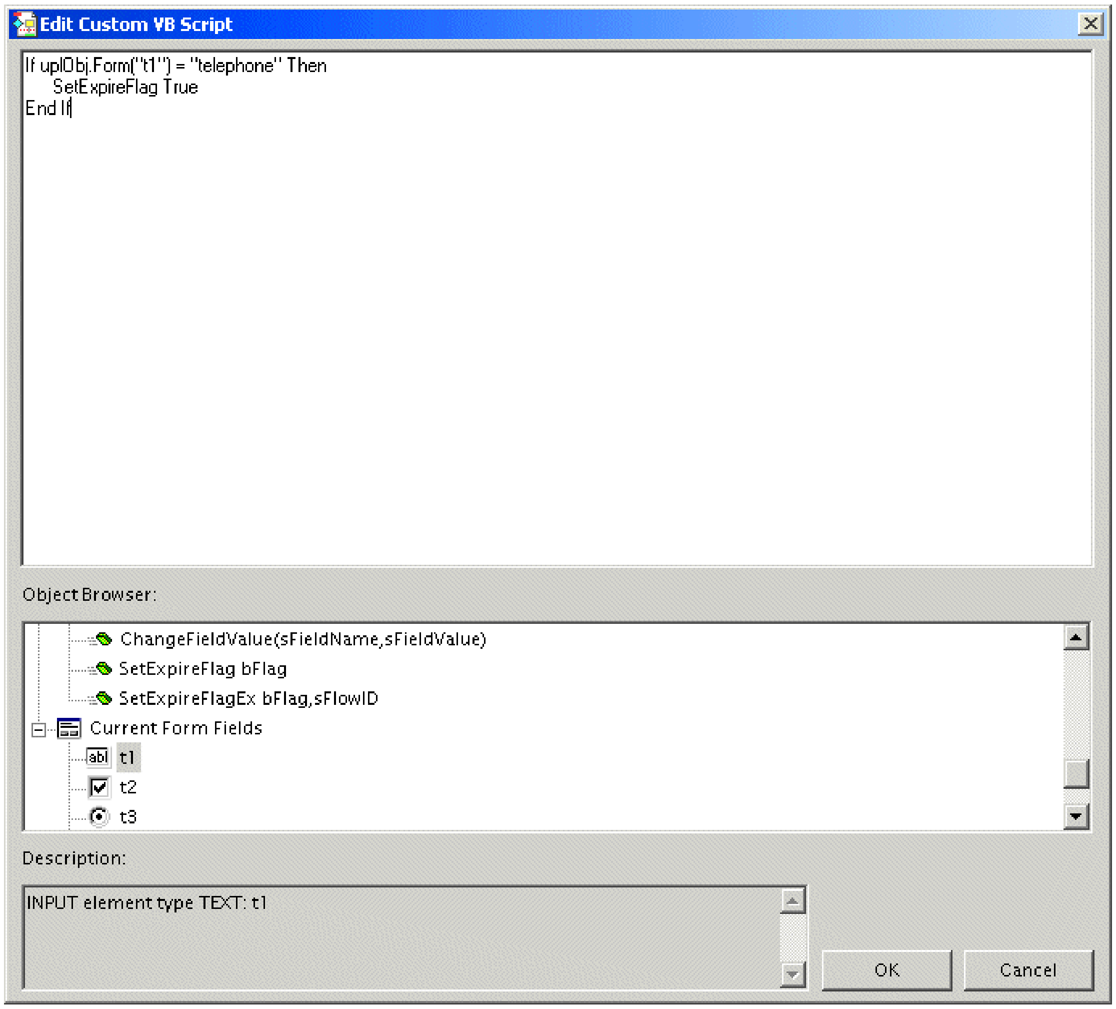


The email dialog expects to see ActiveFlow user Id in the To field and email addresses in Cc list field . When you click the Attachment button it shows files located on the computer that is running ActiveModeler. At run time, the code generated by the rules wizard runs on the computer which has the web server. If the attachment refers to a file on your workstation the attachment may not be found because the file or directory does not exist on the server. To ensure that your attachment specification addresses the correct path you can use a UNC (Universal Naming Code) pathname specification, for example:

\\ComputerName\SomeDirectory\SomeFile.TXT

3. Custom action VBScript

This action allows you compose custom VBScript statements that are executed in-line within the rule action.



```
If uplObj.Form("t1") = "telephone" Then
    SetExpireFlag True
End If
```

INPUT element type TEXT: t1

The Object Browser is very useful to edit Custom VBScript. It displays all the fields from the form attached to the current activity and Active Flow API functions available to write custom scripts.

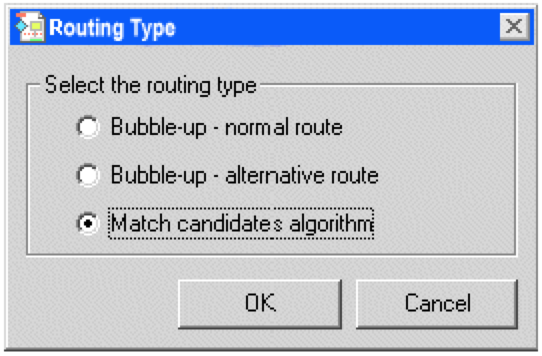
The Active Flow API function or the field name is inserted at the cursor position in the script when double-clicking or pressing enter on an item from object browser.

4. Stop processing all following rules, returning (returned value)

This action causes the rule action to terminate, and according to the returned value , the transaction will progress to the next point in the route or will remain in the user's in-tray.

5. Change routing type to

This action allows you to dynamically change the routing mechanism used by ActiveFlow. You can switch between bubble-up routing and automatic candidate matching.


Routing Type

Select the routing type

( ) Bubble-up - normal route

( ) Bubble-up - alternative route

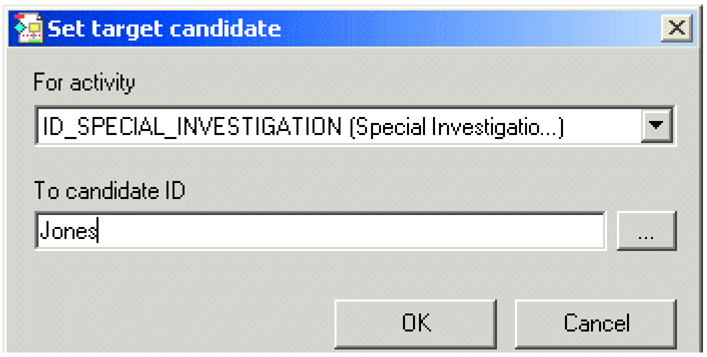
(●) Match candidates algorithm

6. Finish workflow

This action causes the workflow to terminate and the form will be finally approved.

7. Set target candidate

This action allows you to directly specify the person or robotic candidate who will receive the transaction in their in-tray when the current user submits the form. Normally ActiveFlow uses an automatic candidate selection algorithm in which the recipient of the form is chosen by the system. This action allows you to over-ride this mechanism and directly specify a particular candidate.


Set target candidate

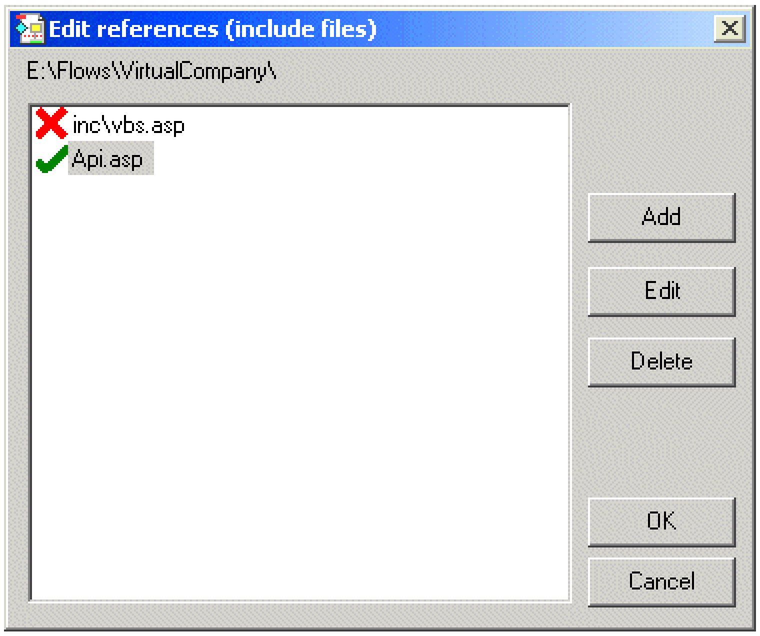
For activity: ID_SPECIAL_INVESTIGATION (Special Investigatio...)

To candidate ID: Jones

You should not over-ride automatic candidate matching as a matter of course, unless you have a very good reason for doing so. This is because the automatic candidate matching system efficiently "load balances" available candidates based on the amount of work that they are doing.

References

Advanced users of ActiveFlow know that the Workflow wizard translates the rules defined with the Rules Wizard into Visual Basic Script. This script is copied into the .asp files generated by the same Workflow wizard for each activity in the map. It is very common practice for workflow designers to add custom VB Script code into rules for highly customizing their applications. This additional script is usually common for more than one activity. In such case is very useful to organize this reusable script in function and subroutines stored in files instead of copying them in each rule.
The files that contain reusable script can be included into rules by using the **References** button in the main dialog of rules wizard. You can **Add**, **Edit** and **Delete** references with the available buttons. A reference is a relative path to a file that contains reusable script. The paths are relative to the **ASP output directory,** which is specified in the **ActiveFlow settings...** dialog. For example:



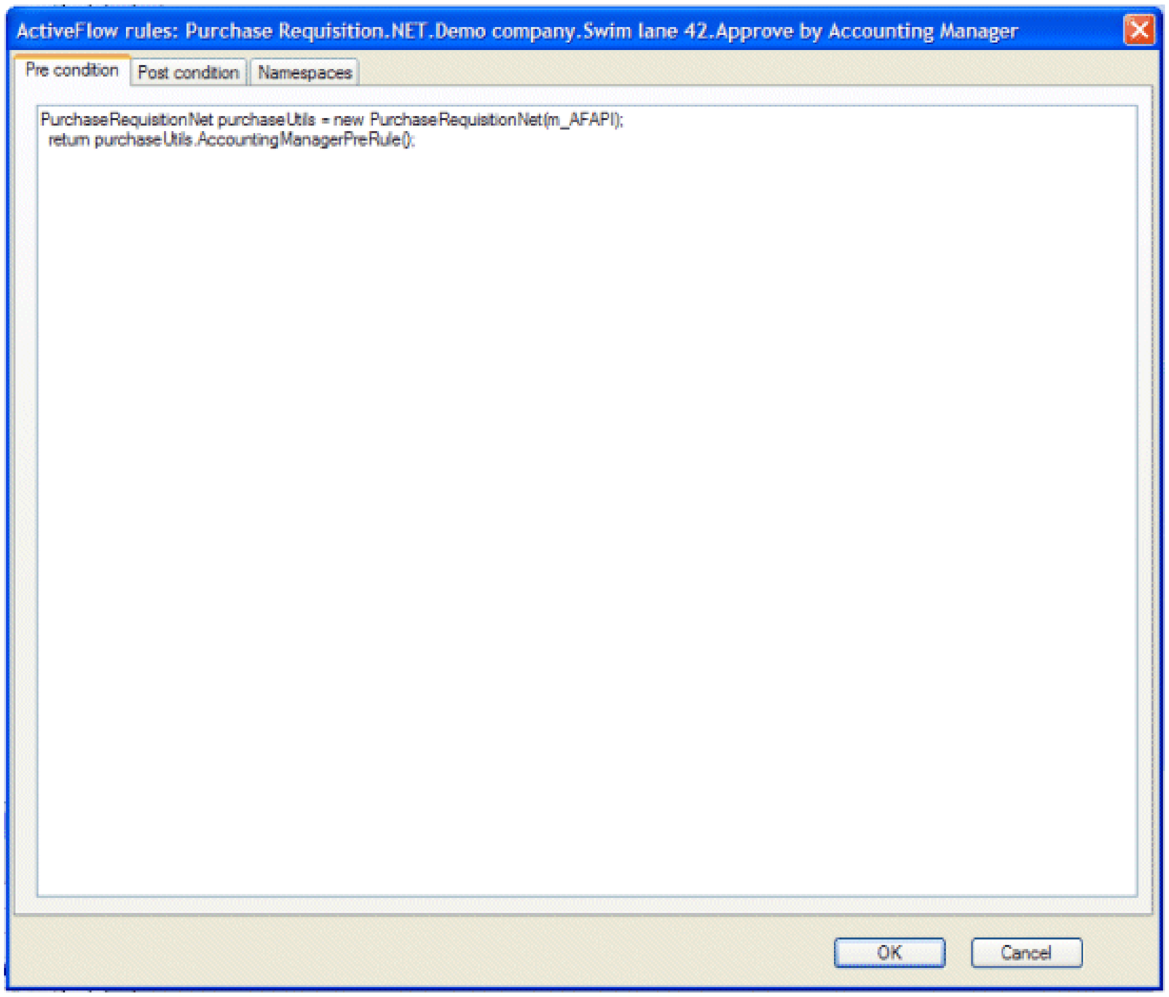
For each reference added, an **Include** directive is added at the beginning of the generated .asp file. This means that in case of the example above, the following lines are added to the generated .asp file.

```
<!-- #INCLUDE FILE="inc\vbs.asp" -->
<!-- #INCLUDE FILE="Api.asp" -->
```

The icons at the left of each reference indicates whether the path and filename specified are valid or not. In our example, the green check sign means that the referenced file exists at the "E:\Flows\VirtualCompany\Api.asp", and the red sign means that the reference doesn't exists at the "E:\Flows\VirtualCompany\inc\vbs.asp" and we should copy the vbs.asp file at the correct location or delete the reference.

RULES WIZARD DIALOG FOR ASP.NET WORKFLOWS

The rules wizard dialog associated to the .NET workflows displays 3 tabs: 2 for defining pre and post condition actions and one for defining the .NET namespaces. The workflow designer must write C# code in this tabs.


ActiveFlow rules: Purchase Requisition.NET.Demo company.Swim lane 42.Approve by Accounting Manager
Pre condition
Post condition
Namespaces
OK
Cancel

```
PurchaseRequisitionNet purchaseUtils = new PurchaseRequisitionNet(m_AFAPI);
 return purchaseUtils.AccountingManagerPreRule();
```

PRE CONDITION

The workflow designer should use the PreCondition function for defining the code which is to be executed before ActiveFlow transaction starts e.g.: check ERP system for budget/stock availability, etc. Also the PreCondition function should contain the rule for routing the form in the case of an OR-Split or Complex-Split. The function must return a string representing the ID of the active output link. If there are several active output links (such as in the case of the Complex-Split), their IDs must be separated by <,>.

Post condition

Post condition function is called by the ActiveFlow framework after it completed the ActiveFlow transaction. The workflow designer can use this function for finalizing operations to external systems e.g. update the budget/inventory on the ERP system, send notification messages, etc.

The post condition function must return **true** if the ActiveFlow should commit the transaction and **false** otherwise.

Namespaces

The Namespaces tab can be used by the workflow designer for declaring classes which can be used from the pre and post condition functions.

Note: A good practice is to define all the business rules or operations in external objects which would only be called then from the pre or post condition rules.

Important: The ActiveFlow APIs are implemented in a class called ActiveFlowAPI and they can be accessed via the variable named AF_API.

Example: Below is an example of custom code which can be written in order to route the form according to certain rules and then to update the external ERP system with several form values.

The assumption is that we have 2 classes MyWorkflowRouting and MyERPUpdate which implement the business functionality and that the files MyWorkflowRouting.cs and MyERPUpdate.cs have been added to the **Custom rules code** table in the project settings dialog.

In the *Namespaces* tab we have to define:

```
using MyWorkflowRouting;
using MyERPUpdate;
```

Pre condition code sample

```
MyWorkflowRouting oWkfRt = new MyWorkflowRouting();
String sFormValue =  AF_API.GetFormValue("FormField", false);
String sActiveLinks =  oWkfRt.ComputeActiveLinks(sFormValue);
return sActiveLinks;
```

Post condition code sample

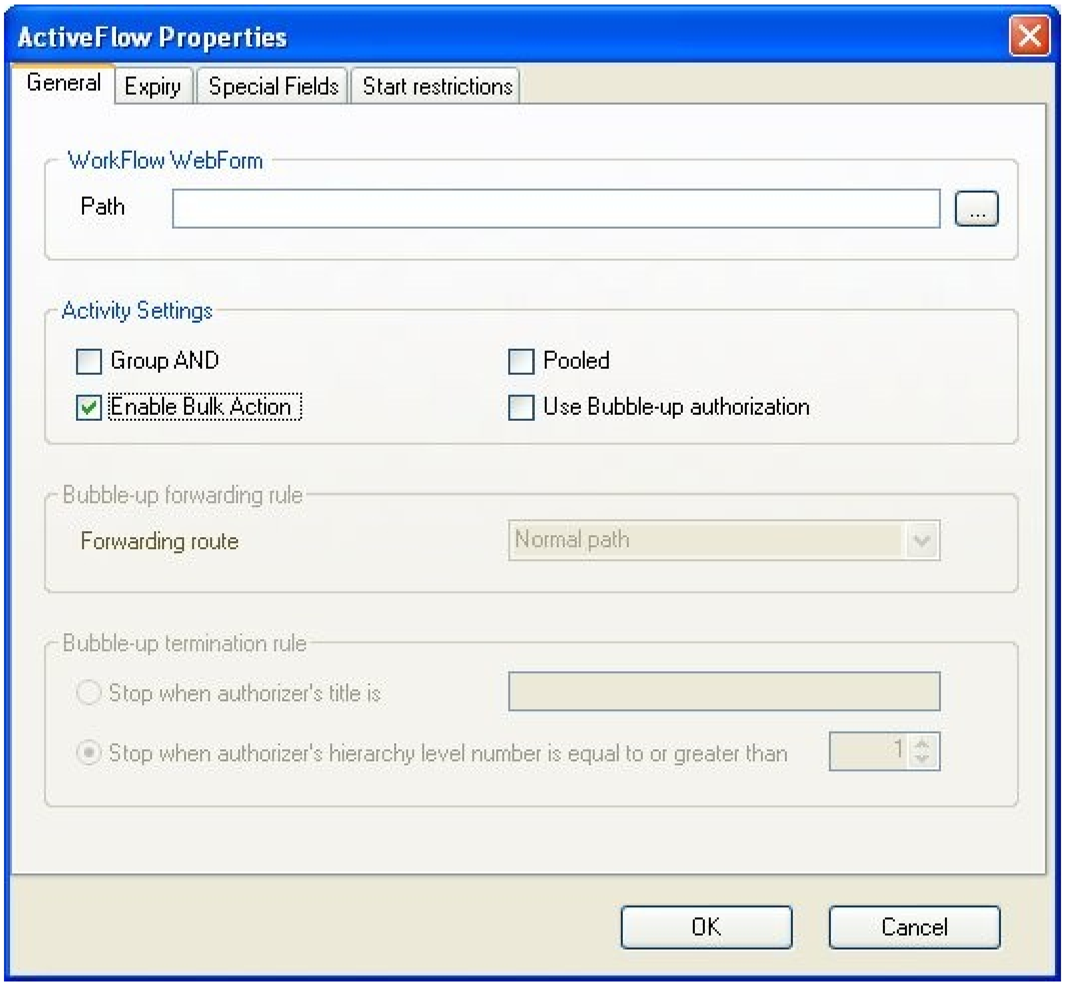
```
MyERPUpdate oERPObj = new MyERPUpdate();
String sFormValue1 =  AF_API.GetFormValue("FormField1", false);
String sFormValue2 =  AF_API.GetFormValue("FormField2", false);
if( oERPObj.Update(sFormValue1, sFormValue2))
        return true;
else
        return false;
```

Bulk action

It is often not appropriate to examine each workflow item "one by one".
For instance, an Accounting department may not need to check each expense workflow individually as approval has already been granted by the issuing departmental manager. However, the accounting department may need to check certain key fields and these can appear in the line item. Approval can be for the whole screen, for selected line items and the actual whole form can be examined (and optionally approved or rejected if required) by clicking on the line item.

The ActiveFlow "Bulk action" feature enables the above scenario. A workflow designer can easily enable the Bulk action functionality for an activity. This means an end user can examine for approval/rejection all the forms associated with that activity in a line item bulk mode.

In order to enable the Bulk action attribute for an activity, right click on the activity in ActiveModeler Avantage, and select **ActiveFlow-> Settings...** . In the **ActiveFlow Properties** dialog box check the **"Enable Bulk Action"** checkbox.


ActiveFlow Properties

General | Expiry | Special Fields | Start restrictions

WorkFlow WebForm
Path:

Activity Settings
- [ ] Group AND
- [ ] Pooled
- [x] Enable Bulk Action
- [ ] Use Bubble-up authorization

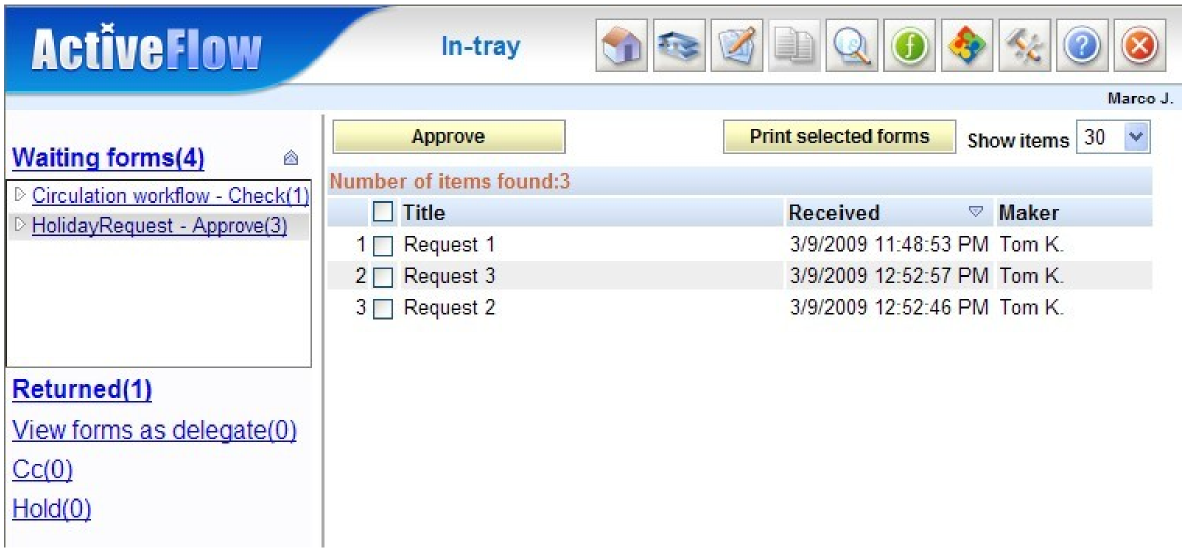
Bubble-up forwarding rule
Forwarding route: Normal path

Bubble-up termination rule
- ( ) Stop when authorizer's title is
- (●) Stop when authorizer's hierarchy level number is equal to or greater than 1

OK | Cancel

Example:

From the ActiveFlow In-tray screen, select the ***Waiting forms*** category on the left side of the screen. Click the double arrow icon at the right side of the ***Waiting forms*** to expand the list. Select the workflow activity from the list. The list of items for Bulk action is then displayed as shown in the image below:



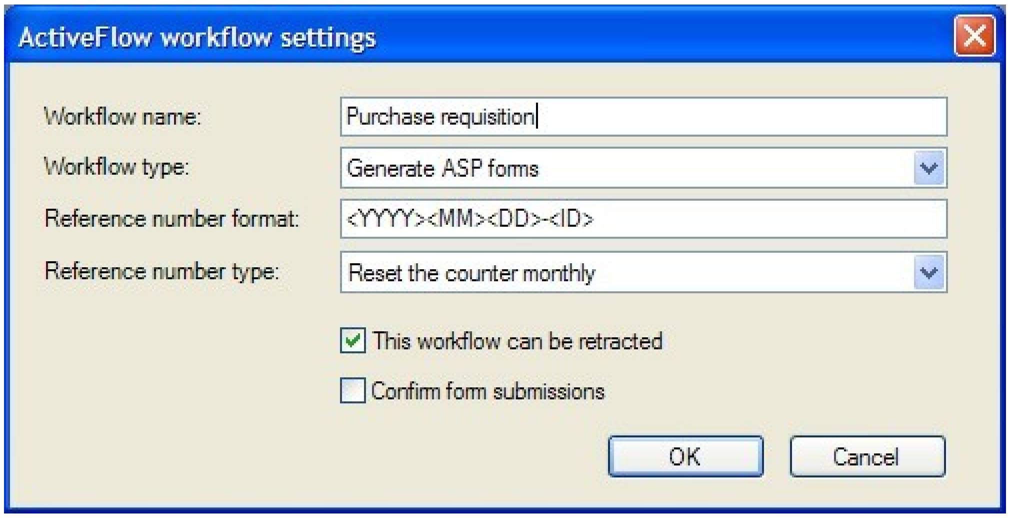
Number of items found:3

	Title	Received	Maker
1	Request 1	3/9/2009 11:48:53 PM	Tom K.
2	Request 3	3/9/2009 12:52:57 PM	Tom K.
3	Request 2	3/9/2009 12:52:46 PM	Tom K.

When displaying the list of items for Bulk action, ActiveFlow can also display in read-only mode field values from the form. Please check the Special fields section in order to find out how to specify these.

Cancel workflow

This functionality allows a maker to cancel a previously issued workflow. This is a diagram associated setting and in order to enable it, right-click on the diagram in Workspace Navigator and select **Settings...** under the **ActiveFlow** menu.



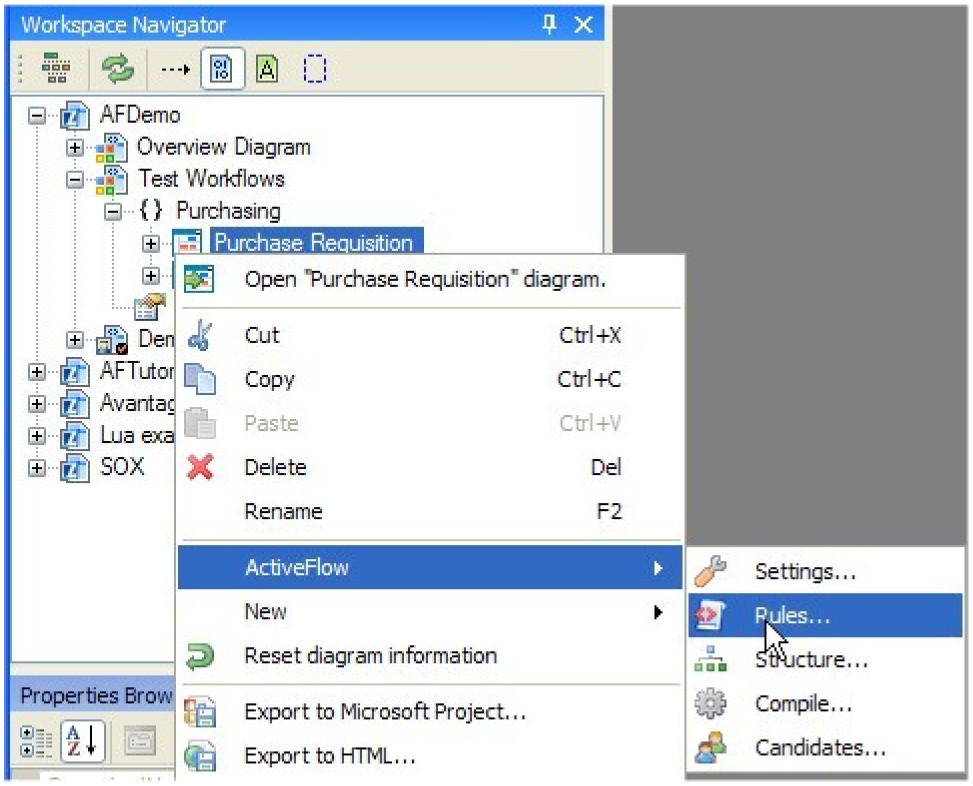
Workflow name:	Purchase requisition
Workflow type:	Generate ASP forms
Reference number format:	<YYYY><MM><DD>-<ID>
Reference number type:	Reset the counter monthly

[✓] This workflow can be retracted
[ ] Confirm form submissions

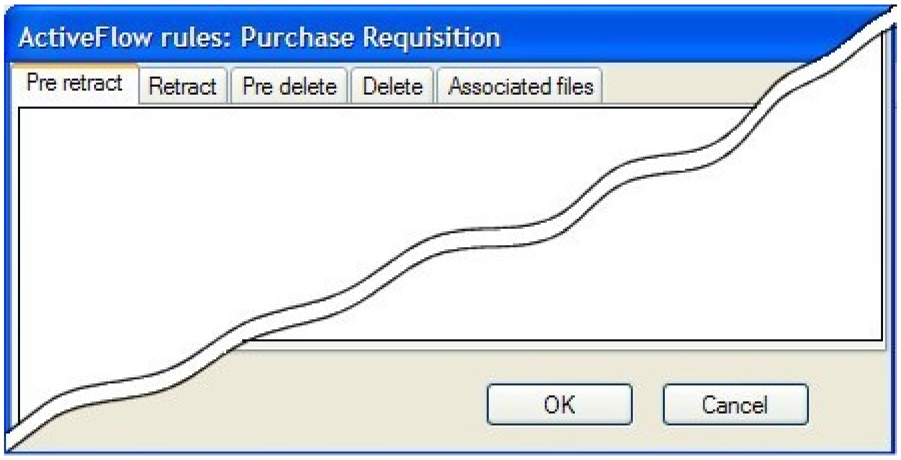
While canceling a workflow you can choose to retain the original form in the hold list.

If specific actions have to be performed in the case a workflow is canceled or deleted from the hold list (e.g. Update records in external databases) ActiveFlow provides places where you can write your own custom code.

For this right click on the diagram and select **ActiveFlow->Rules**



According to the workflow type (ASP or ASP.NET), the custom code written in the retract and delete functions should be VBScript or C# respectively.



PreRetract -should contain code which will be executed before the ActiveFlow engine actually deletes data from the ActiveFlow database. Workflow values may be retrieved by using the ActiveFlow API function AF_GetSentFieldValue. The function must return True or False. If the function returns **False** the ActiveFlow will abort the cancel workflow action.

PostRetract - should contain code to be executed after ActiveFlow engine deletes data from the ActiveFlow database. The function must return True or False. If the function returns **False** the ActiveFlow will undo all the changes in the ActiveFlow database putting all the deleted records back.
Note: At this stage data does not exists in the ActiveFlow database so the function AF_GetSentFieldValue will not work.

PreDelete - should contain code which will be executed before the ActiveFlow delete the data for the held form.

PostDelete – should contain code to be executed after ActiveFlow engine deltes data for the held form.

Associated files - the designer may use this section for including files or defining global variables, include files etc. The global variables can be used for passing values from PreRetract to PostRetract functions, or from PreDelete to PostDelete functions.

Example VBScript for ASP forms:

Pre retract tab

```
<%
nTotalValue =  AF_GetSentFieldValue("TotalValue")
bResult = CancelRequest(nTotalValue)
if(bResult) then
	OnPreRetract = True
else
	bNotifyClients = False
	OnPreRetract = False
end if
%>
```

Retract tab

```
<%
if(bNotifyClients) then
	SendNotificationMailToVendor
end if
%>
```

Associated files tab

```
<!--#INCLUDE FILE=”MyCustomCode/AccountingAPI.asp”-->
<!--#INCLUDE FILE=”MyCustomCode/NotificationsAPI.asp”-->
<%
dim bNotifyClients
bNotifyClients = True
%>
```

Example C# for ASP.NET forms:

Pre retract tab

```
Int32 nTotalValue = 0;
bool bResult =  false;
AccountingAPI oAccObj = new AccountingAPI();
nTotalValue =  m_AFAPI.AF_GetSentFieldValue(“TotalValue”);
bResult = oAccObj .CancelRequest(nTotalValue);
if(bResult){
        MyGlobalVariables .bNotifyVendor = true;
        return true;
}else{
        MyGlobalVariables .bNotifyVendor = false;
        return false;
}
```

Retract tab

```
NotificationsAPI oNotifObj = new NotificationsAPI();
if(MyGlobalVariables .bNotifyVendor)
        oNotifObj .SendNotificationMailToVendor;
return true;
```

Associated files tab

```
using Accounting;
using Notifications;

class MyGlobalVariables{
        public static bool bNotifyVendor = false;
}
```

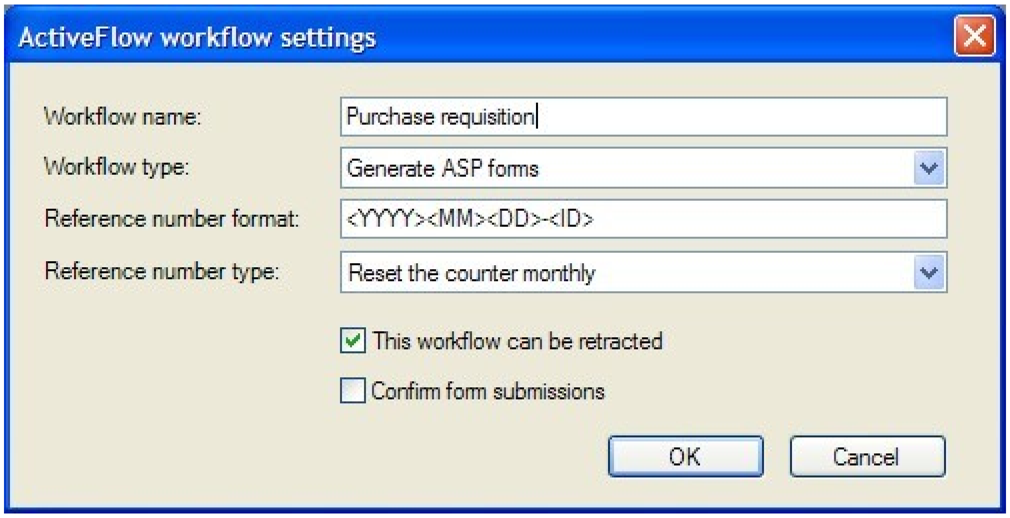
Note:

- If the maker cancels a workflow, all the ActiveFlow users who previously actioned the form (approved, returned) will be notified by mail.

- It is not possible to cancel a workflow once it has been archived (finally approved or rejected).

Reference number

If every workflow instance should have an identification number you can use the reference number functionality. The refrence number should be set for every diagram. To specify the format of the reference number right-click on the diagram in Workspace Navigator and select **Settings...** under the **ActiveFlow** menu:



Workflow name:	Purchase requisition
Workflow type:	Generate ASP forms
Reference number format:	<YYYY><MM><DD>-<ID>
Reference number type:	Reset the counter monthly

[✓] This workflow can be retracted
[ ] Confirm form submissions

The reference number format can be any string and can include the following predefined tags (the < > characters are mandatory):

Predefined Tags	Value
<CSTR>	A string that is set from the content of the form field named "_AFRefNo_String".
<DD>	The current day of the month. If the day is less than 10, the day is displayed with a leading zero.
<ID>	The counter of workflows. It is incremented every time a workflow is submited.
<ID,x>	The counter of workflows formated on minimum x characters. If the counter have less than x digits, leading zeros are added until the minimum width is reached.
<MM>	The current month of the year. If the month is less than 10, the month is displayed with a leading 0.
<YY>	The current year without the century. If the year without the century is less than 10, the year is displayed with a leading zero.
<YYYY>	The current year with the century.

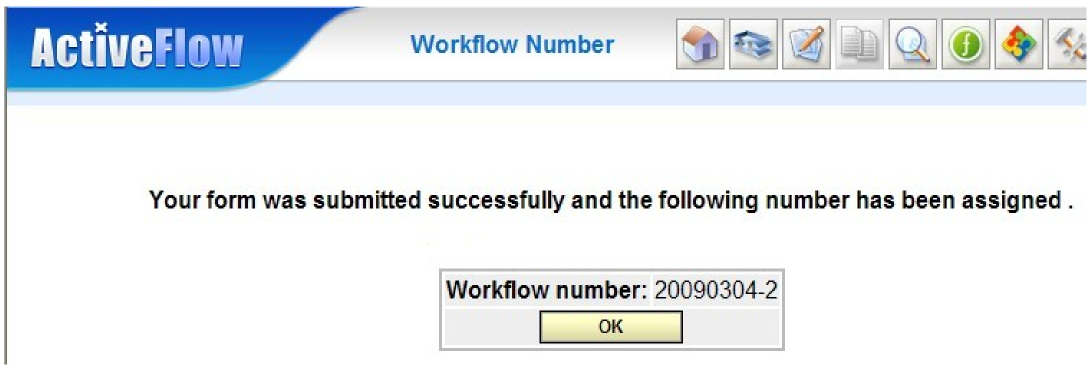
Example

If the counter is 1202 and the submission date is 03/17/2009 and the specified format is <YYYY>/<MM>/<DD>-ABC-<ID> the generated number is 2009/03/17-ABC-1202.

From the reference number combo box it can be selected the type of the reference number:

- None - the refrence number is disabled
- Do not reset counter - the counter is not reset automatically
- Reset the counter montly - the counter is reset at the beginning of each month.
- Reset the counter yearly - the counter is reset at the beginning of each year.
- Custom reset - the reset date of the counter will be set by custom code

If the type of the is enabled, after submiting a workflow the generated reference number is displayed in browser.



Your form was submitted successfully and the following number has been assigned .

Workflow number: 20090304-2

Special fields

Quite often it is necessary to make a search in the ActiveFlow database depending on a field value in the form. For example, a financial controller might want to see all the approved Travel Expenses forms with a total amount greater than 250 USD.

In order to implement such a functionality, ActiveFlow has the concept of "Special fields".

The Special fields can be used in 2 different ways:

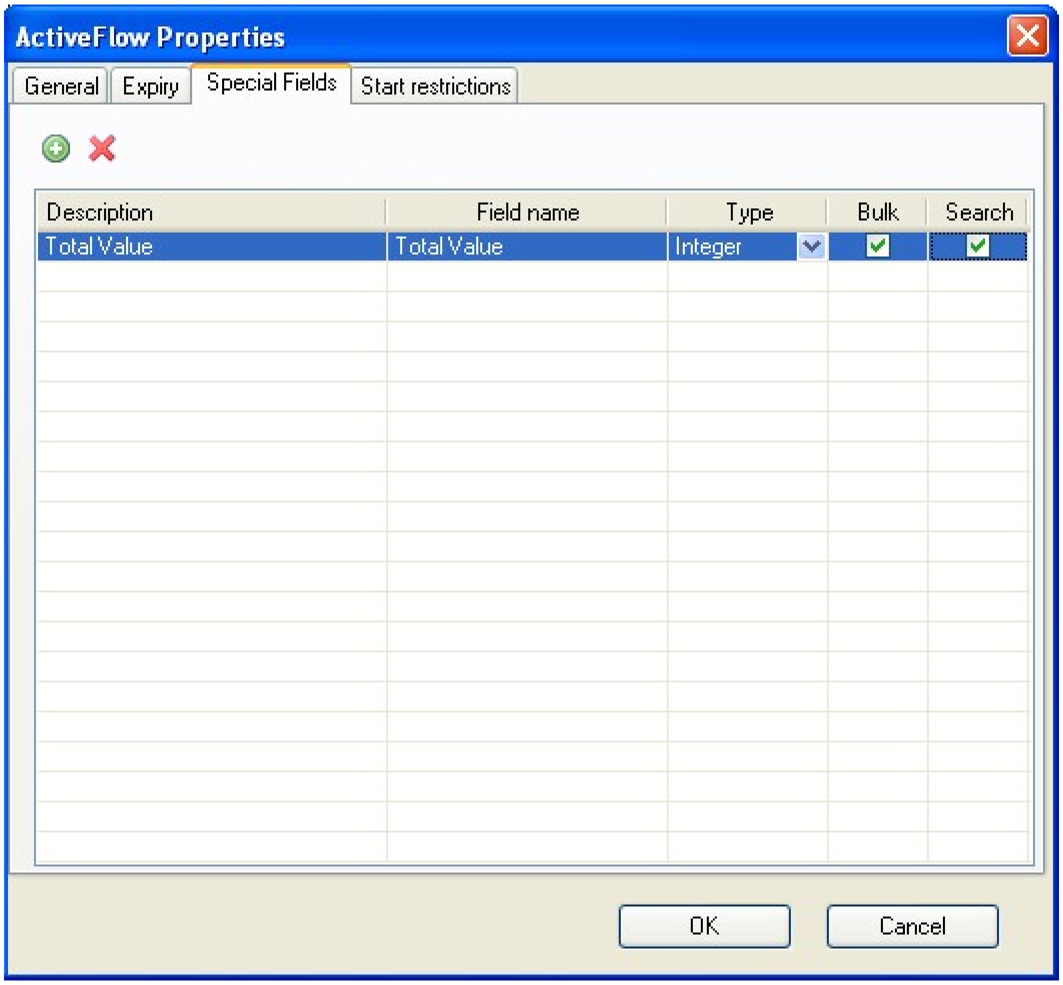
- In the Bulk action approval list from the In-tray page. A user can see the values in the form without opening each form which is a very useful productivity feature.
- As a search restriction in the In-tray and Enquiry pages

The Special fields can be defined using the ***Special fields*** tab from the **ActiveFlow Properties** dialog (right click on an activity and choose ***ActiveFlow->Settings...***).

In the Special fields properties dialog, the designer will need to specify the following:

Special Field attribute	Value
Description	A meaningful description of the field. The description will be displayed as heading in the Bulk approval list and in the list of Special fields in the search by value section of the In-tray or Enquiry pages.
Field Name	The name of the field from the form attached to the activity.
Type	The type of the value stored in that field (if the field may contain strings and numbers, set it to string).
Bulk	Checked if the special field is to be used for displaying values in the Bulk approval list.
Search	Checked if a search can be performed based on this field.

In the example below, the designer defined a searchable field. Also the user selects to perform Bulk action, so the values from this field will be displayed in the In-tray page.

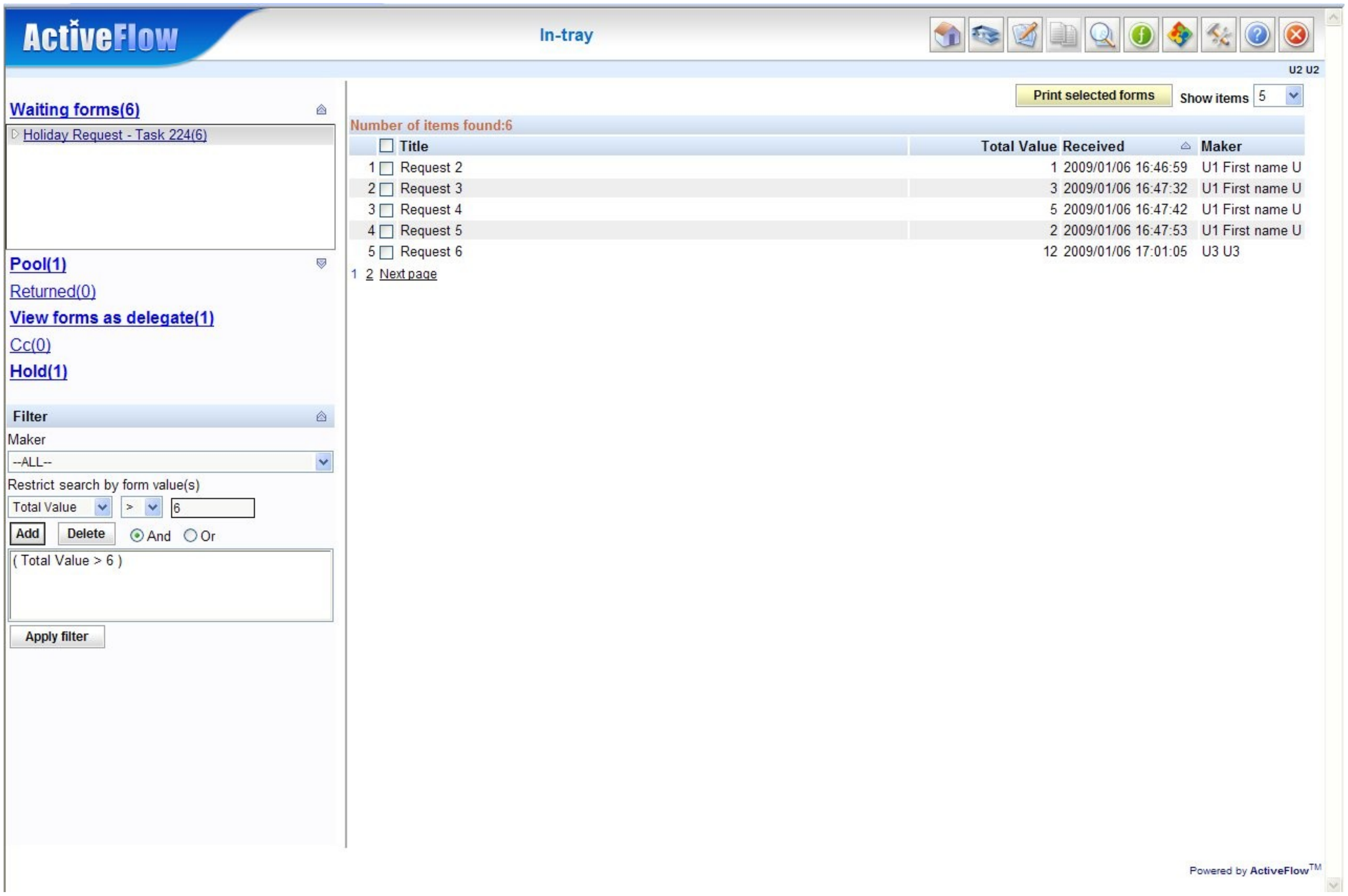


Description	Field name	Type	Bulk	Search
Total Value	Total Value	Integer	✓	✓

The Special fields might be used in the:

- In-tray page from ActiveFlow

Click on the double arrow icon at the right side of the ***Waiting forms*** to expand the list. Select the workflow from the list.



ActiveFlow In-tray

U2 U2

Waiting forms(6)

Holiday Request - Task 224(6)

Pool(1)

Returned(0)

View forms as delegate(1)

Cc(0)

Hold(1)

Filter

Maker: --ALL--

Restrict search by form value(s): Total Value > 6

Add Delete [✓ And] Or

(Total Value > 6)

Apply filter

Print selected forms Show items 5

Number of items found:6

	Title	Total Value	Received	Maker
1	Request 2	1	2009/01/06 16:46:59	U1 First name U
2	Request 3	3	2009/01/06 16:47:32	U1 First name U
3	Request 4	5	2009/01/06 16:47:42	U1 First name U
4	Request 5	2	2009/01/06 16:47:53	U1 First name U
5	Request 6	12	2009/01/06 17:01:05	U3 U3

1 2 Next page

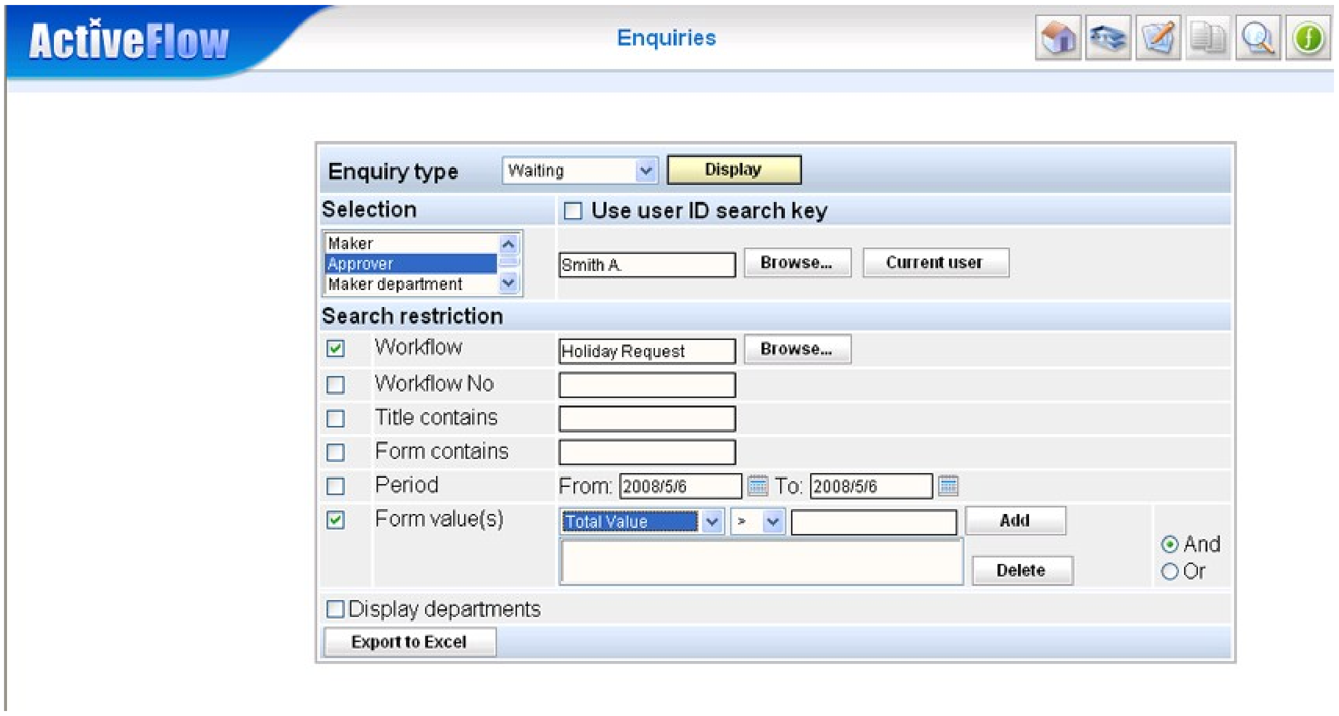
Powered by ActiveFlow™

The items for Bulk action and the “Total Value” special field are displayed on the right side of the page.

If you want to further restrict the list of forms currently displayed, you can filter the list using the "Form field value" filter. This is available only for waiting and pool type forms.

- Enquiry page from ActiveFlow

The Special fields might be used in the Enquiry page as a search restriction.



Enquiry type	Waiting	Display
Selection	☐ Use user ID search key	
Maker / Approver / Maker department	Smith A.	Browse... Current user
Search restriction		
☑ Workflow	Holiday Request	Browse...
☐ Workflow No		
☐ Title contains		
☐ Form contains		
☐ Period	From: 2008/5/6 To: 2008/5/6	
☑ Form value(s)	Total Value >	Add Delete [✓ And] Or
☐ Display departments		
Export to Excel		

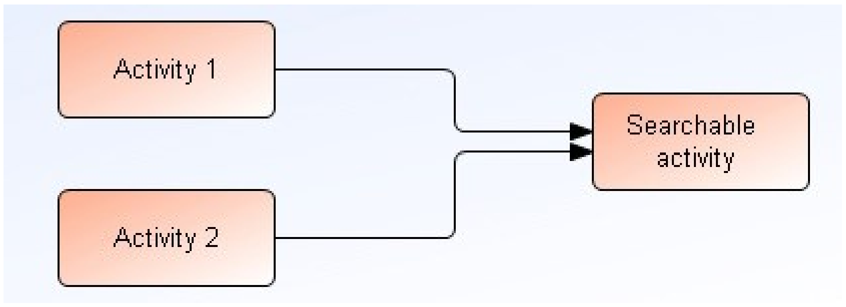
Select the Enquiry type field and a workflow in the ***Workflow*** field. After that you can choose a search restriction in the ***Form values(s)*** field.

Notes:

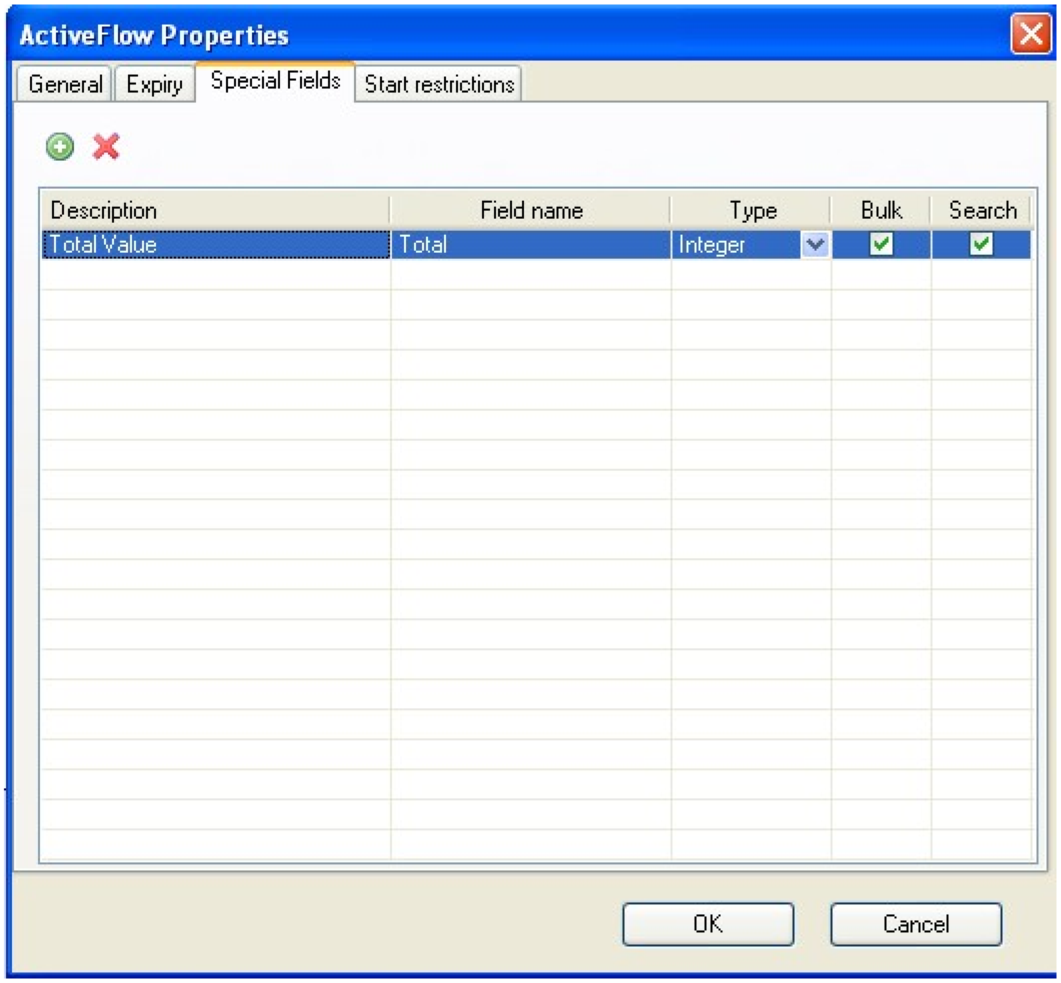
1) As Special fields are associated with an activity, they can be used (in the In-tray or Enquiry pages) only if an activity/workflow is selected.

2) If the **Bulk** checkbox is checked for a property in the **ActiveFlow Properties** dialog, it means the associated field will be displayed in the Bulk In-tray for that activity.

3) In order to use the **Search** function in the In-tray and Enquiries page the designer has to define the searchable fields for all incoming activities.



In the above example, if the designer wants to enable a search by ***Total*** for the activity called "**Searchable activity"** in the *In-tray* and Enquiries, the designer must add Special fields properties for each of the incoming activities: **Activity 1** and **Activity 2** as below:



Description	Field name	Type	Bulk	Search
Total Value	Total	Integer	✓	✓

IMPORTANT NOTE !

Defining the searchable fields for an activity means that the user can search for those values **sent** from that activity.

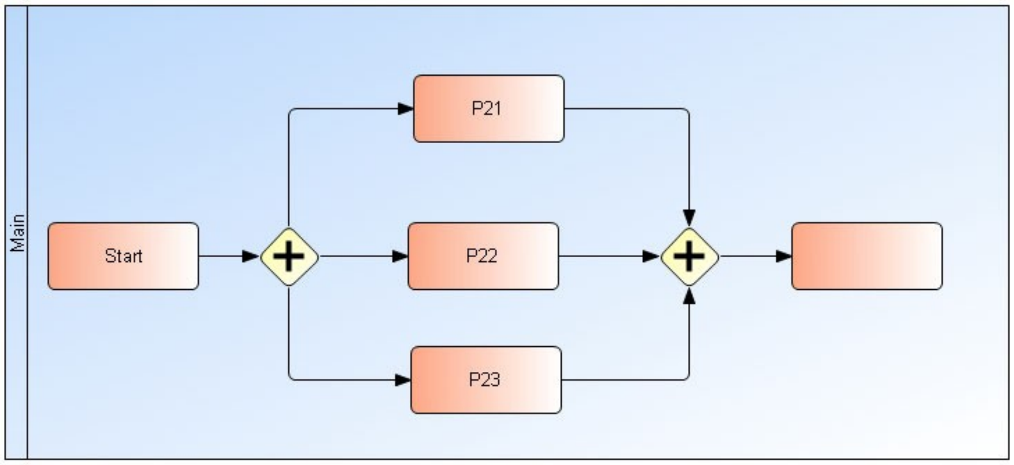
Expiry settings

The Activity Expiry function enables control over when an activity should be handled (approved/returned/rejected).
The expiry date may be specified by the designer at workflow design-time or by a user at run-time when submitting the form.
When a job expires, an ActiveFlow component will automatically try to perform an action specified by the workflow designer. The component will try to perform this action several times, waiting between retries a specified number of hours and after that it will try to perform a

second action.
The actions which can be specified are :

- Send a warning email
- Approve the job
- Return to maker
- Return to previous
- Reject

Also, ActiveFlow provides a mechanism (API) to temporarily disable expired jobs for a workflow. This is particulary useful in the case of an AND-Split when it is a requirement that the parallel jobs can expire only after at least one of the jobs is approved.



Design Time

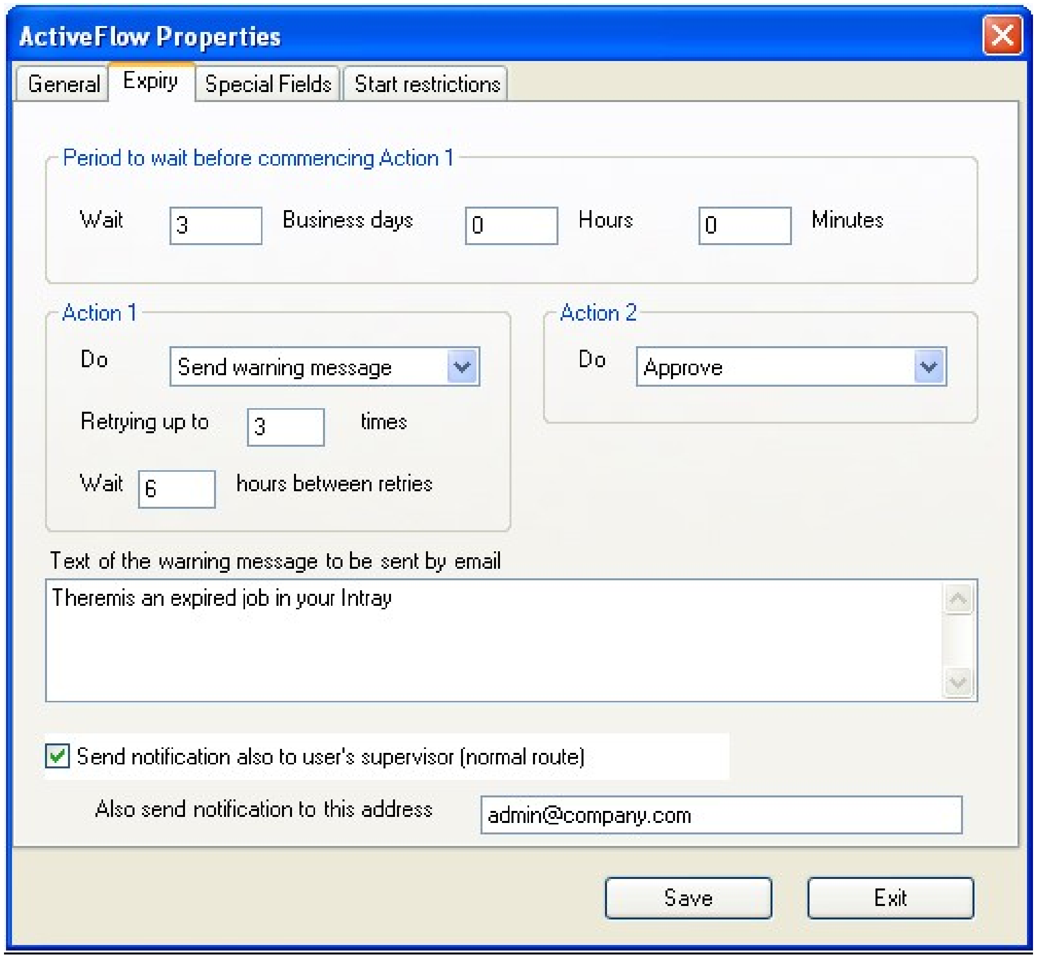
In the above example a map is made with the expiry function set on.
A special requirement is made that before expiry, at least one of the parallel activities must approve.

In this case, when the maker issues a workflow from the Start activity, the workflow designer may disable the expiry functionality for this particular workflow in the Transition condition function. Therefore for this particular workflow, when any of the activities P21, P22, P23 are approved, the workflow designer may enable the expire functionality so the remaining jobs will expire according to the settings in the ActiveModeler. For more details please see SetExpireFlag function in the API section.

The ActiveFlow component which handles the expired jobs is an service which runs in background and from time to time checks the ActiveFlow database.

Specifying the expiry date at "design-time" :

For each activity in a map, the workflow designer may specify the expiry date and a set of actions which will be executed when the job will expire.
In ActiveModeler, use the **Workflow expiry settings** dialog (right-click on an activity and select the **ActiveFlow/Settings...** menu) the designer will specify the values required for this functionality.



Period to wait before commencing Action 1

Wait 3 Business days 0 Hours 0 Minutes

Action 1

Do: Send warning message

Retrying up to 3 times

Wait 6 hours between retries

Action 2

Do: Approve

Text of the warning message to be sent by email

Theremis an expired job in your Intray

[✓] Send notification also to user's supervisor (normal route)

Also send notification to this address: admin@company.com

Note:

- If the designer specifies 0 as the number of business days, hours and minutes the job will expire as soon as it will enter the user's In-tray.
- Action1 will be performed at least once so if the designer specifies the number of retries as 3, Action1 will be performed 4 times (at most if the job isn't handled in the meantime).
- If an activity is not allow to expire, the Action1 should be set to *None* and number of business days, hours and minutes to 0 (these are the default values).
- if Action1 is "Send warning message" the designer may specify also to send a notification mail to the user's supervisor (normal route) and/or to a certain email address.

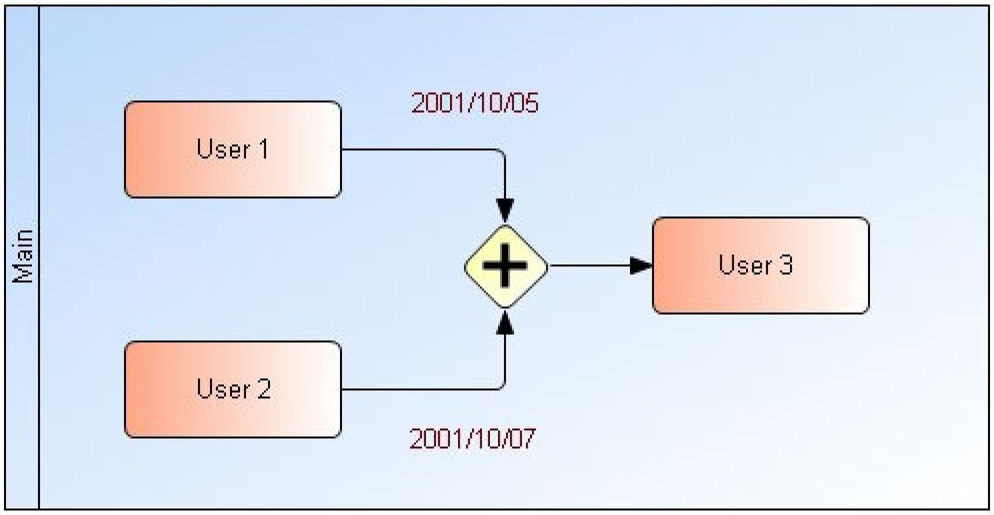
The workflow designer/administrator has to specify the special holiday calendar using the **Administrative forms** function of ActiveFlow. Monday to Friday is counted as the working week (business days) and week-ends are holidays.

Specifying the expiry date at "run-time" :

By using the ExpireDate AF Control, the expiry date can be specified at run-time.
Important: please note that this will specify by when the job sent to next user(s) has to be handled and not the date by when the entire workflow should be finally approved.
At run-time, the user can specify only the date by when the next user(s) should handle the job

and not the actions which can be performed. These have to be specified at design-time by the designer.

Note:
1) The expiry date specified at run-time has priority over the value specified at design-time.
Example:
- If the designer specifies for an activity that the associated jobs will expire after 8 business days and the user specifies at run-time (in 2001/10/29) that the job has to be handled by (2001/11/01) then the job will expire in 2001/11/01 at 00:00:00 AM.
2) In the case of an AND-Join, the expiry date will be the most recent date.

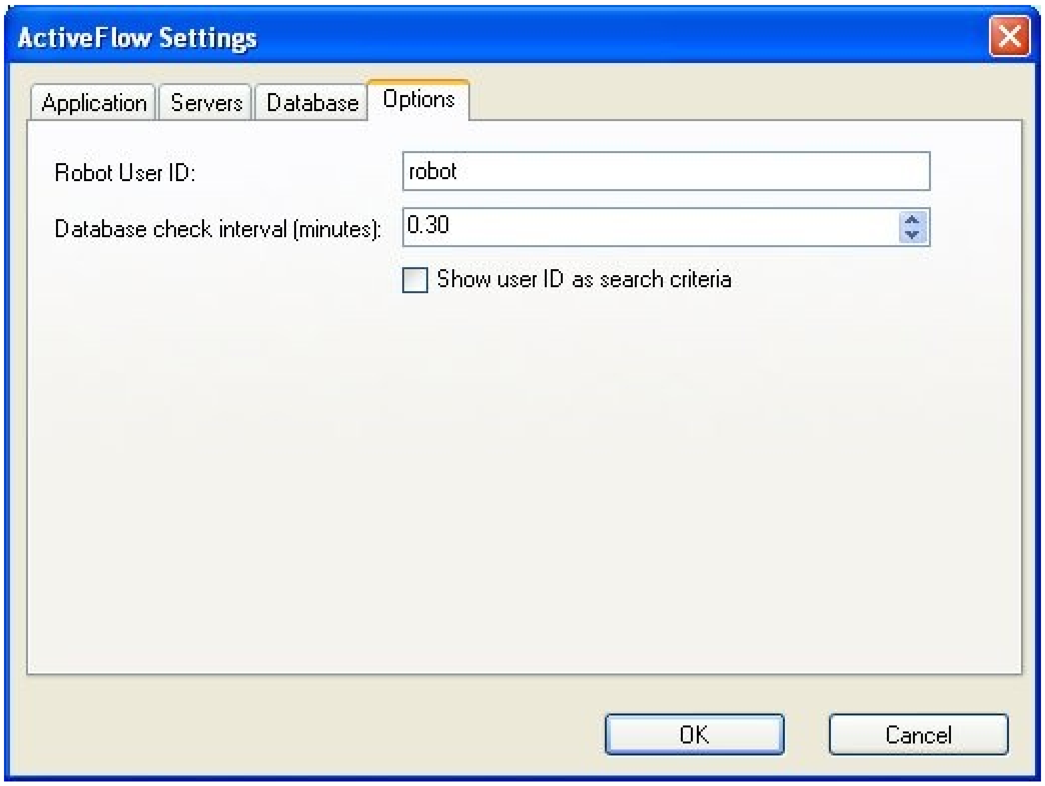


In the above example if user 1 specifies at run-time the expiry date for user 3 as 2001/10/05 and user 2 will specify 2001/10/07, the workflow form for user 3 will expire in 2001/10/07.
3) A returned job cannot expire.

Robot approval

There are cases in a process map when an activity needs to be actioned automatically.
Also, some business flows don't have just one end point and in order to implement a workflow based on them, the designer must add a final activity and set is as a robot activity.

To set an activity as an automatic activity you have to set its candidate user to “robot”and this user should be set as candidate user for all automatic activities across workflow maps. Also, to enable the robot approval functionality right click on the project item in ActiveModeler Avantage and select **ActiveFlow\Project settings...\Options** tab.



Robot User ID: robot

Database check interval (minutes): 0.30

☐ Show user ID as search criteria

In the Robot User ID field specify the robot user ID (usually is "robot") and in the Database check interval specify a value greater than 0.

Note:

The "robot" user is just another ActiveFlow user. The ActiveFlow system will automatically approve all the waiting forms for the user specified in the "Robot userID" field.

Standard notifications

This function allows the workflow designer to set pre-defined actions for each activity. The events for which you can set these pre-defined actions can be grouped into 2 categories: actions triggered when the form reaches the target activity and actions triggered when the user actions a form.

Below is the list of events:

- **Received** – executed when the form reaches the current activity following the approval of the previous activity.
- **Receive returned** – executed when the form reaches the current activity following the return action of one of the subsequent activities.
- **Approve** – executed when the user approves the form attached to the current activity.
- **Return** – executed when the user returns the form attached to the current activity.

- **Reject** – executed when the current user rejects the form attached to the current activity.

For these events you can define the following actions:

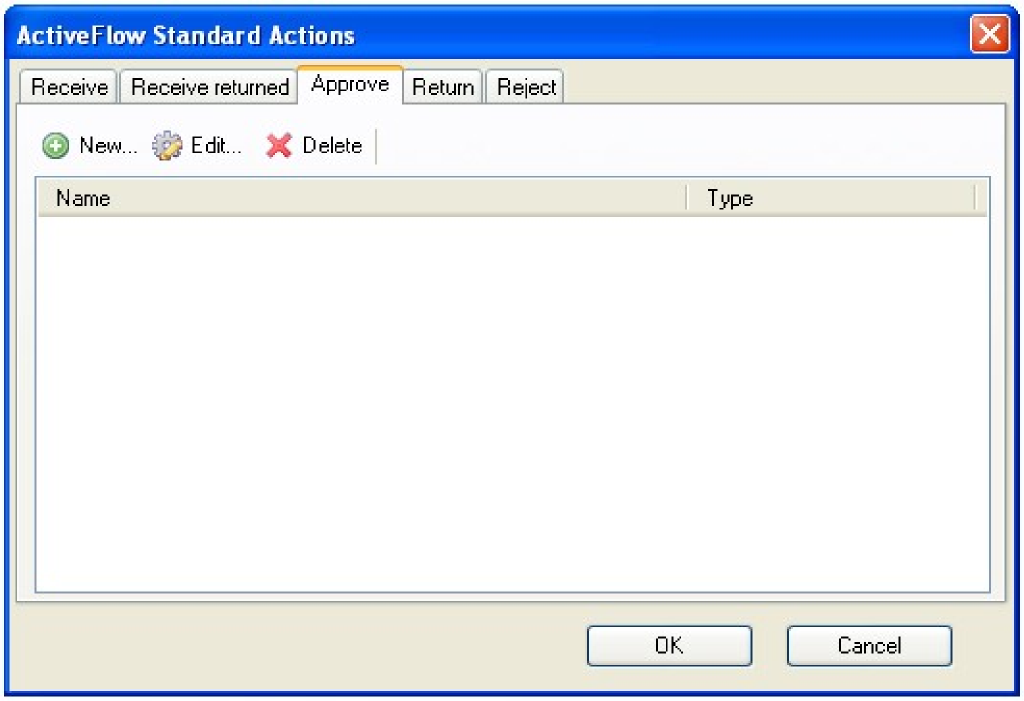
1. Send email / Send direct link email

These 2 actions are similar: both will send a notification email but the second one will also generate internally a token used for user authentication.

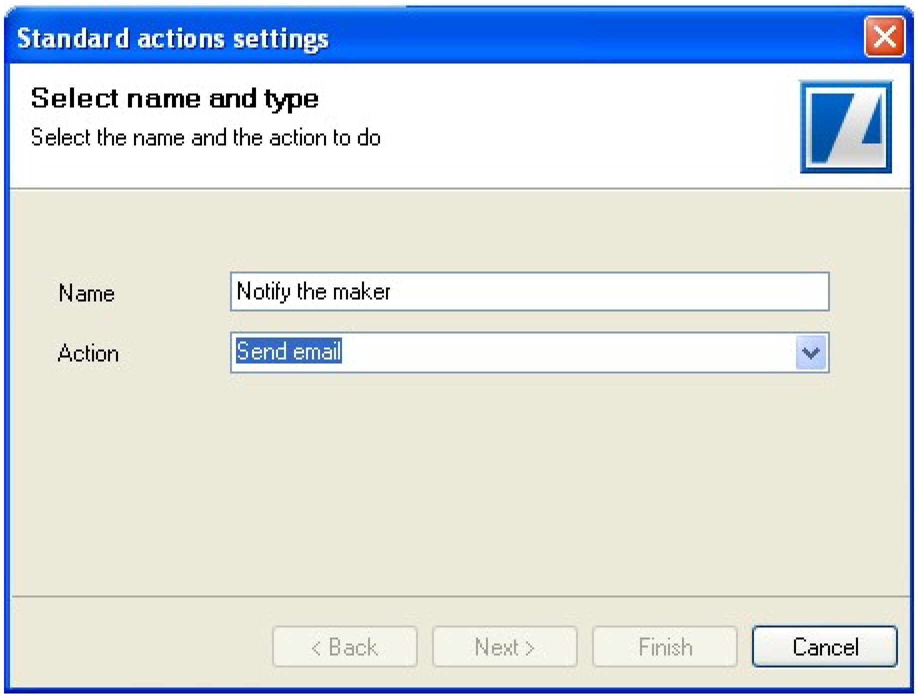
You can customize the content of the mail subject and body and if you want to include certain values you can use the following constructions:

Tag	Description
<AF_VALUE name=fieldName/>	Inserts the value of a form field. E.g. To insert the form title, you should write <AF_VALUE name=_AFFormTitle/> To insert the value of a field called TotalValue you would write <AF_VALUE name=TotalValue/>
<AF_DirectLink_Intranet/>	Inserts a URL which can be used by the Intranet users to access the form. Eg. To create a click here type of link you should write the following: <a href='<AF_DirectLink_Intranet/>'>click here</a>
<AF_DirectLink_Intranet_Integrated/>	Inserts a URL which can be used by Intranet users using integrated authentication to access the form. Eg. To create a click here type of link you should write the following: <a href='<AF_DirectLink_Intranet_Integrated/>'>click here</a>
<AF_DirectLink_Internet/>	Inserts a URL which can be used by Internet users to access the form. Obviously the network settings must allow this approach. Eg. To create a click here type of link you should write the following: <a href='<AF_DirectLink_Internet/>'>click here</a>

To define these standard actions right-click on an activity and select the ***ActiveFlow Standard Actions...*** menu.



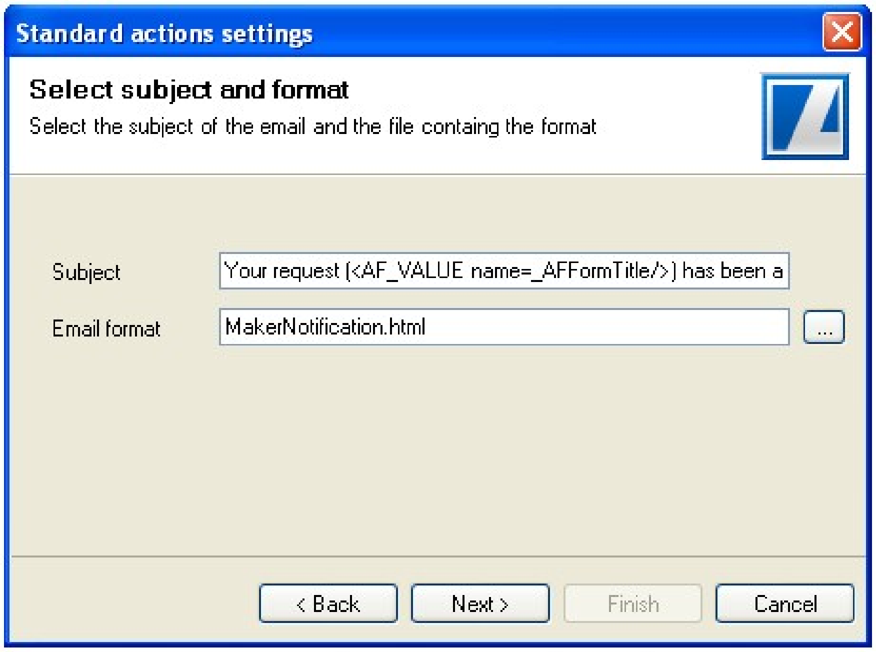
Then select the tab associated with the desired event and press the **New...** button. Enter the action name (it must be unique for an activity) and the actual action.



Select name and type
Select the name and the action to do

Name	Notify the maker
Action	Send email

For sending an email notification the action settings dialog is as below:


Standard actions settings
Select subject and format
Select the subject of the email and the file containg the format

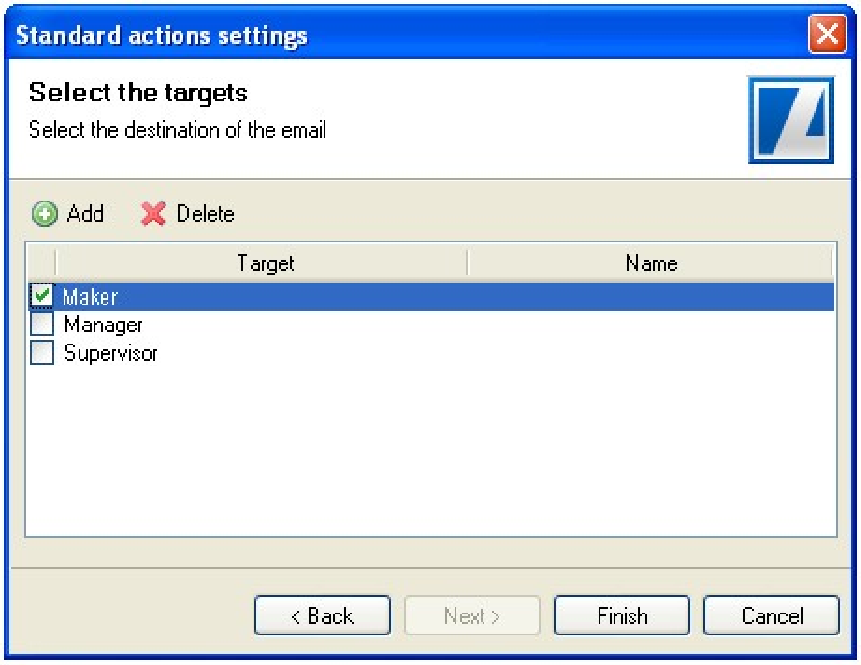
Subject: Your request (<AF_VALUE name=_AFFormTitle/>) has been a

Email format: MakerNotification.html

...
< Back
Next >
Finish
Cancel

You can specify the email subject and also you can specify a form value in it (e.g the form title). The email format is defined as an Html document. This is to allow both to nicely formatted emails using HTML (a text-area control would make the email body hard to control) and to create larger emails.

The next step (and final for the *Send email* action) is to define the destination of the email notification.


Standard actions settings
Select the targets
Select the destination of the email
Add
Delete

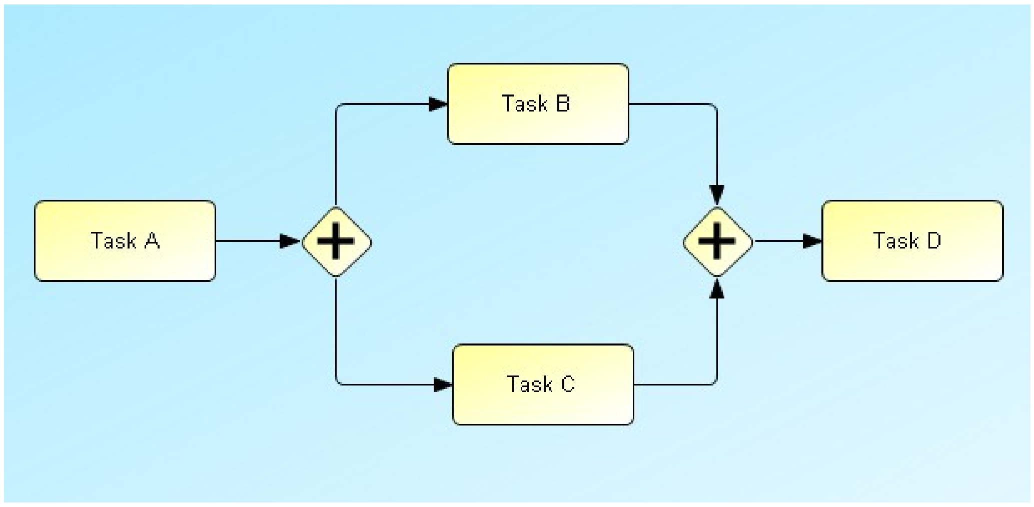
Target	Name
✓ Maker	
☐ Manager	
☐ Supervisor	

< Back
Next >
Finish
Cancel

The table below shows the actions and targets allowed for each type of event/operations.

Event/Operation	Permitted action	Target
Received	Send email	Approver, Approver's delegate, Approver's manager, Approver's supervisor, Custom email
	Send email link	Approver, Custom email
Receive returned	Send email	Approver, Approver's delegate, Approver's manager, Approver's supervisor, Custom email
	Send email link	Approver, Custom email
Approve	Send email	Maker, Current approver's manager, Current approver's supervisor, Custom email.
Return	Send email	Target user(s), Custom email.
Reject	Send email	Maker, Current approver's manager, Current approver's supervisor, Custom email.

Example: Let's consider the example below:



If you want to notify the the users of tasks B and C that they got a new form when the user submits the form at TaskA, you have to select both tasks (Task B and Task C) and add an action *Send email* for the event *Received* having as target the *Approver*.

On the other side, if you want to notify the user for tasks B and C when the user returns the form at Task D, you have to define the action *Send email* for the action *Return* at activity Task D and set as target the *Target users*. Another option is to define the action *Send email* for the event *Receive returned* for tasks B and C and setting as target the *Approver*.

This design might look confusing to some workflow designers but it allows a greater flexibility such as to send the email notification only to the user handling Task B.

E-Mail notification

The ActiveFlow engine will automatically notify the involved users in the case of the following actions:

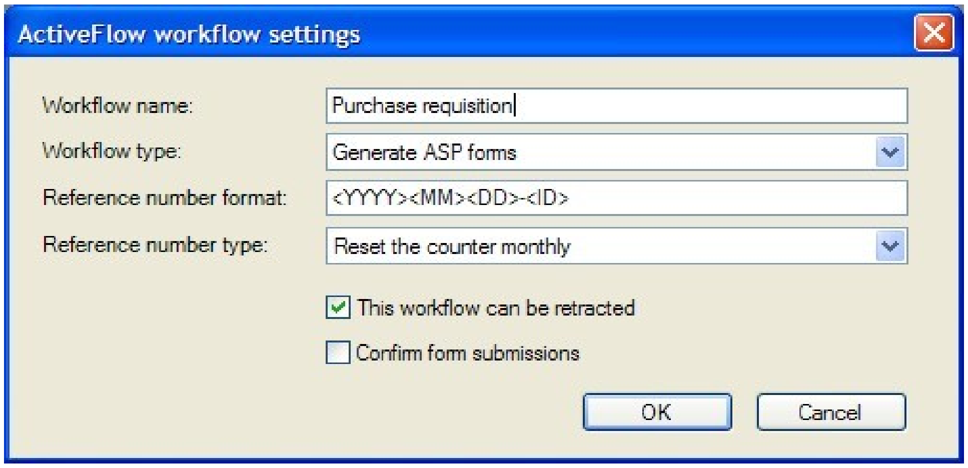
- delegate action
- emergency action

The email contains the following information.

- Date
- First name and last name of the user who actioned the form
- Workflow name
- Form title
- Comments in the case the form has been rejected or returned

Workflow structure and special view settings

Using ActiveModeler you can define a workflow tree to group your workflows processes. Before defining the workflow structure it is necessary to set the workflow properties in the workflow settings page. Right-click on the diagram in Workspace Navigator and select **Settings...** under the **ActiveFlow** menu.


ActiveFlow workflow settings

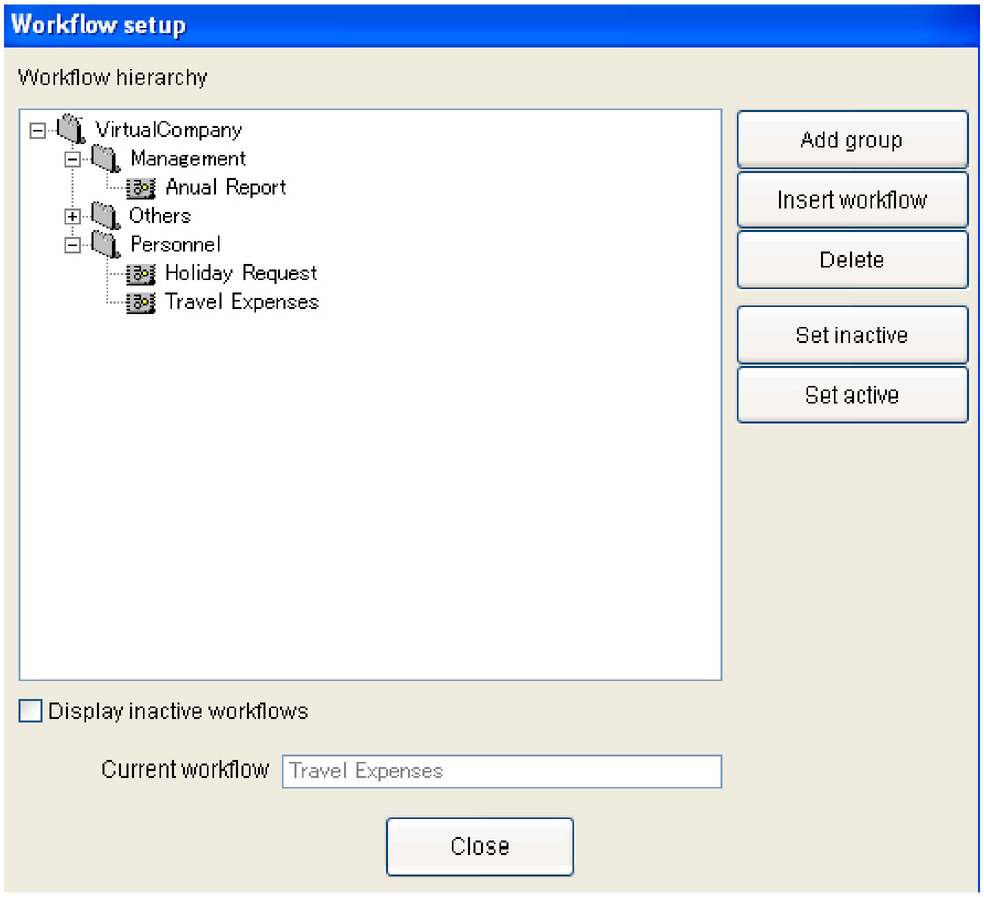
Workflow name: Purchase requisition
Workflow type: Generate ASP forms
Reference number format: <YYYY><MM><DD>-<ID>
Reference number type: Reset the counter monthly
[✓] This workflow can be retracted
[ ] Confirm form submissions

The settings are explained below:

Workflow name	The name of the specific workflow within the application.
Workflow type	The type of the workflow (ASP or ASP.NET).
Reference number format	The reference number format. For more details check the Reference number section.
Reference number type	The reference number type. For more details check the Reference number section.
This workflow can be retract	If checked the form in this workflow can be retract by the maker. For more details please check the Cancel workflow section.

Confirm form submissions	If checked before any submit a standard comfirmation messages is displayed.

Having the workflow name defined is time to place it within the workflows hierarchy. To do so right-click on a diagram in Workspace Navigator and select **Structure settings...** under the **ActiveFlow** menu.



This dialog allows the designer to organize workflows into groups, to delete existing workflows or to insert the current workflow into the workflow hierarchy. The top of the workflow hierarchy is the application. This top item is not editable and not removable. If a workflow is not used any more, it should be inactivated. When you do so, it won't be displayed in the start new workflow page. The inactive workflows have different icons.
Please note that only the current workflow may be inserted.

Important!

If a workflow which contains in-flight transactions is deleted from the workflow hierarchy, all the in-flight forms will be inaccessible and the data displayed in the In-tray page and enquiries will be inconsistent. The forms will become visible only after the deleted workflow is inserted back in the workflow hierarchy and after running the workflow wizard for the correct map.
The same phenomenon will occur if the designer deletes an activity for which there are in-flight forms. In this case, all those forms will be inaccessible until an activity with the same activity ID as the deleted one is placed back in the map and the workflow wizard run again.

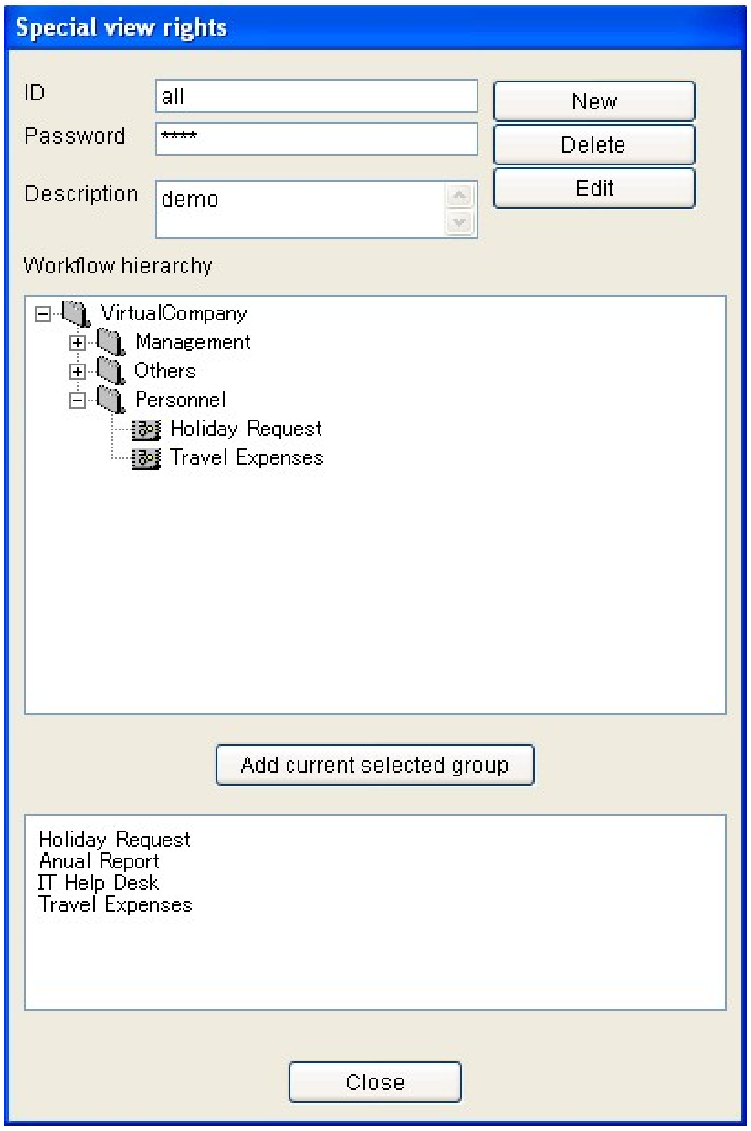
Note because the workflow setup dialog is not modal, if the user activates another application before the above dialog is displayed, the dialog window will be active in the background.

Special view

The ActiveFlow standard enquiries are based on the rule:

- A user can see anything he/she submitted/approved
- A user can see anything a person from the same department submitted/approved if that person's hierarchy level is lower (0 is the highest level) than current user's hierarchy.

The ActiveFlow special view functionality extends the above rule allowing the workflows to be grouped under an ID protected by a password. A user has unrestricted view access to any workflows from this group provided that the user knows the special password.



ID: all
Password: ****
Description: demo

VirtualCompany
Management
Others
Personnel
Holiday Request
Travel Expenses

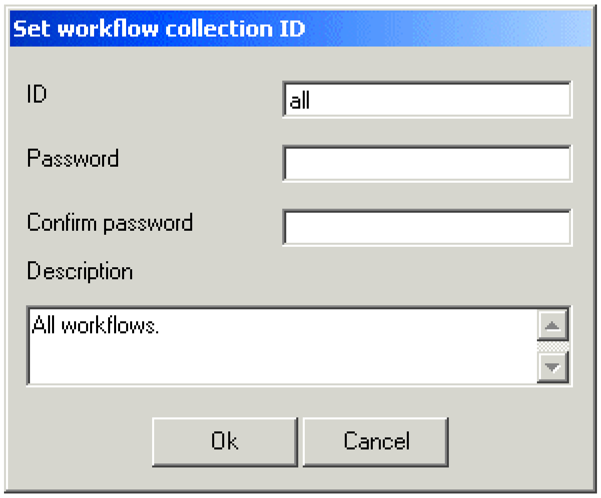
Holiday Request
Anual Report
IT Help Desk
Travel Expenses

Defining special views

Create a new special view ID:

The special view groups can be defined from the **ActiveModeler Avantage**. Right click on the project node, select **ActiveFlow** -> **Special View**… menu.

For viewing the workflows in a certain group, the designer must enter the ID and password and then press ENTER. Double-clicking a workflow from the workflow hierarchy will add the workflow in the active collection and double-clicking an item in the workflow list will remove it from that collection.
The *New*, *Delete* and *Edit* buttons on the upper right side of the dialog allow the user to manage (add new, change description, password, delete) the collection IDs.


Set workflow collection ID

ID: all

Password:

Confirm password:

Description: All workflows.

.

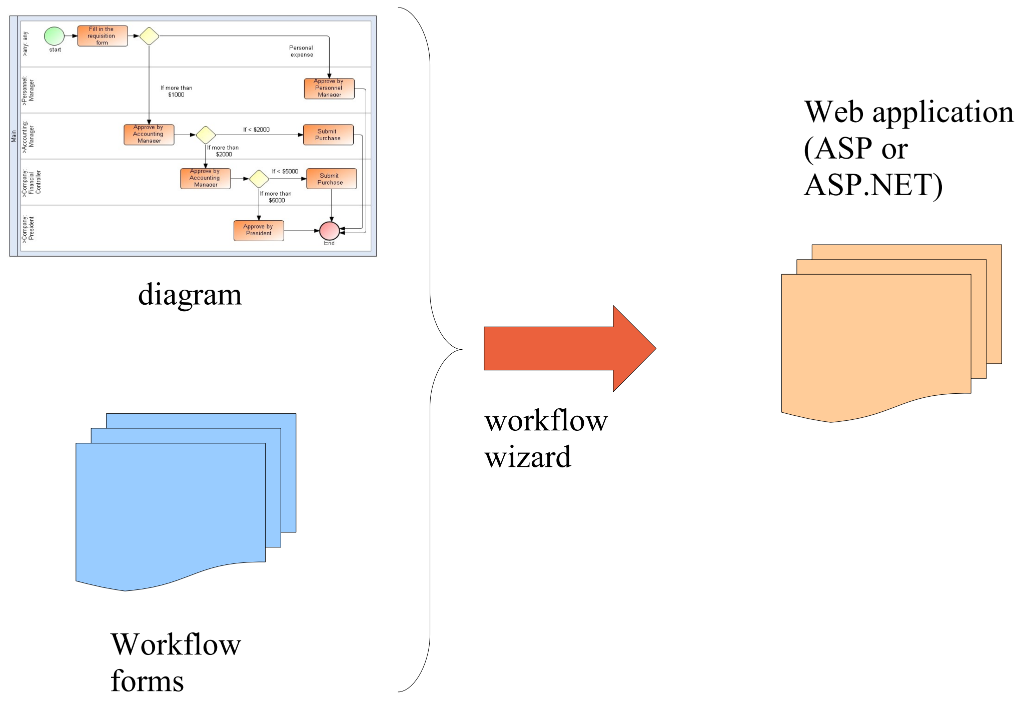
Using special views

In the activeflow web application , Special function menu , select the Special view page. Enter a valid special view ID and password created as per the above procedure.

Workflow wizard

An ActiveFlow workflow is an ASP or ASP.NET web application based on the framework provided by the ActiveFlow engine.

This web applications are created by a component of ActiveModeler Avantage called workflow wizard. Based on the workflow diagram and the workflow forms associated to the activities, the workflow wizard generates the ASP or ASP.NET files necessary for running the workflows.



Note: After every change which affects the business logic in the workflow diagram (modify the routing rules or paths, change the activities' properties, etc), changes in the workflow forms or modifications in the custom files, you have to re-run the workflow wizard for the affected diagram in order to re-generate the web application files.

To run the workflow wizard you have to right-click on the diagram and then select **ActiveFlow/Compile...** menu. Prior to running the wokflow wizard for any diagram you must have completed the following steps: create the ActveFlow database and set the ActiveFlow properties in the **ActiveFlow\Project settings...** dialog.

Workflow customization

ASP-based workflows

Workflows using HTML/ASP forms are quite easy to customize: external code/functions can be defined in a separate file (E.g. MyCustomCode.asp) placed under the web application folder (preferably under a separate folder) and then this file can be either included in the workflow form

<!--#INCLUDE FILE="CustomCode/MyCustomCode.asp"-->

either set as reference in the rules wizard dialog (see the Rules wizard section for more details) either set as a global include file in the project settings dialog (see the Project settings/Application tab section for more details).

ASP.NET-based workflows

In the case of using ASP.NET framework, you can use your own .NET objects in the Active Flow forms/rules. For this you can just place your objects in the following pre-defined folders

- **App_Code** - App_Code folder can contain standalone files that contain code to be shared across several pages in your application).

or

- **Bin** - The Bin folder is like the App_Code directory, except it can contain precompiled assemblies. This is useful when you need to use code that is possibly written by someone other than yourself, where you don't have access to the source code (VB or C# file) but you have a compiled DLL instead. Simply place the assembly in the Bin folder.

Sample Classes:

Create a new class, and copy it in the App_Code folder.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
public class CustomClass
{
        public CustomClass(){
        //
        // TODO: Add constructor logic here
        //
        }
        public String CustomMethod(){
                return "CustomMethod called at: " + DateTime.Now.ToString();
        }
        public String GetCustomValue(){
                Random rn = new Random();
                return rn.Next(1000000).ToString();
        }
}
```

Create a new class library, compile it, and copy it in the Bin folder

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace CompiledCustom{
public class CompiledCustomClass{
        public String CompiledCustomMethod(){
                return "CompiledCustomMethod called at:" + DateTime.Now.ToString();
        }
}}
```

Using the custom classes

1. Using the custom code in forms

Add a text box control in the aspx form:

```
<asp:TextBox ID="AFTextBoxSample" runat="server"></asp:TextBox>
```

Add the following code in the “code behind” file

```
protected override void CustomCode_AfterOnLoad(){
        CustomClass cstClass = new CustomClass();
        AFTextBoxSample.Text = cstClass.GetCustomValue();
}
```

The function GetCustomValue of the CustomClass returns a random number so each time the form is loaded the textbox will be have a different value (random).

2. Using the custom code in workflow rules

Add the following code in the rule editor (Pre condition or Post condition):

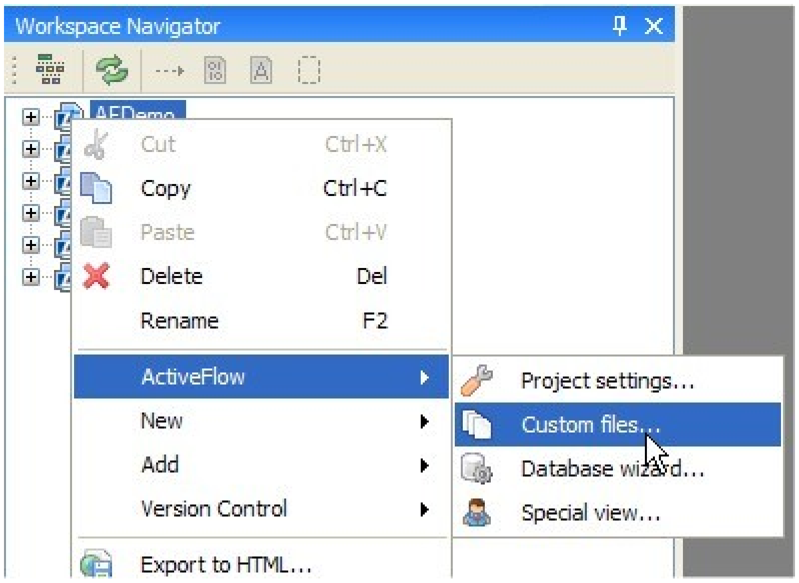
```
CustomClass cstClass = new CustomClass();
CompiledCustom.CompiledCustomClass compiledCustomRule = new
       CompiledCustom.CompiledCustomClass();
String tmpResult;
tmpResult = cstClass.CustomMethod();
AF_API.WriteToLog("testCustomCode", tmpResult);
tmpResult = compiledCustomRule.CompiledCustomMethod();
AF_API.WriteToLog("testCustomCode", tmpResult);
return "true";
```

The result of the methods CustomMethod and CompiledCustomMethod is written in the "testCustomCode_yyyymmdd.log" file under ActiveFlow\Bin\Log folder.

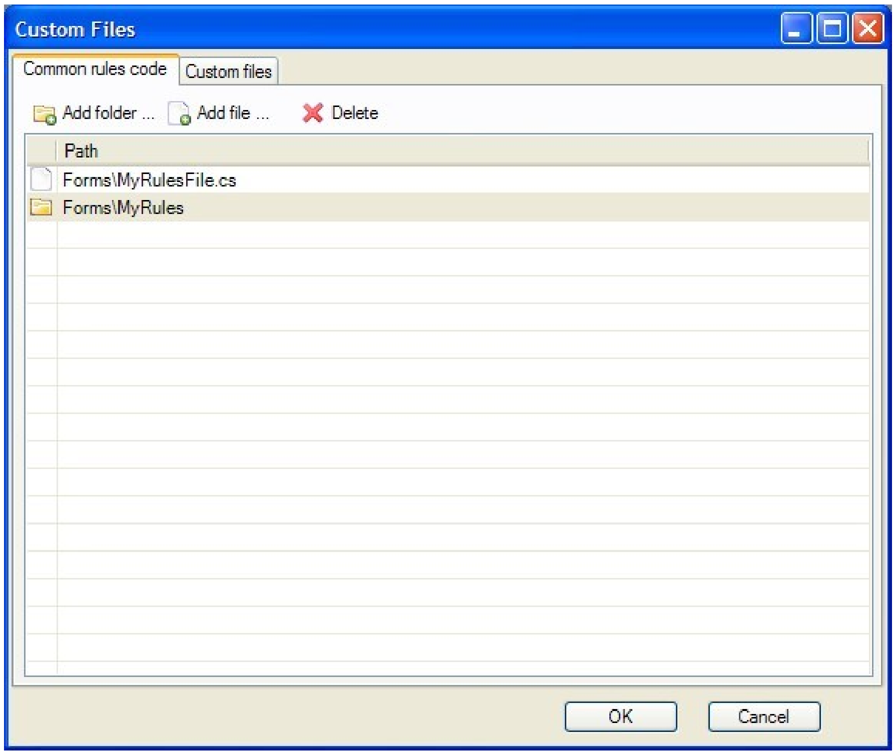
Custom files

The custom files dialog helps working with your custom code files. Every time a workflow is compiled all the files are copied in the appropriate location.

Right-click the project item and select **ActiveFlow->Custom files**...



The opened dialog have two tabs: Common rules code (files set here are copied in the Application directory/App_code) and Custom files (files set here are copied directly in the Application directory). If a folder is added in any of the custom files category all the content of the folder is copied including the subfolders.



Path
Forms\MyRulesFile.cs
Forms\MyRules

ActiveFlow extensions

Batch Admin Toolset

Many companies want administrative change to come directly from a central corporate source, such as the Personnel Department, with interfaces to the various systems that require this change information. In certain situations, administrators may prefer not to use online administration tools directly because doing so can be time-consuming if there are many admin requests. Also important is that batch admin from a central source avoids repeated work on different systems if all systems have an automated admin function.

The Batch Admin Toolset (BAT) makes ActiveFlow administrative operations more efficient. The following 12 administrative operations can be achieved with BAT:

1. Load departments
2. Delete departments
3. Rename departments
4. Move departments
5. Load roles
6. Delete roles
7. Rename roles
8. Move roles
9. Load users
10. Delete users
11. Move users
12. Modify users

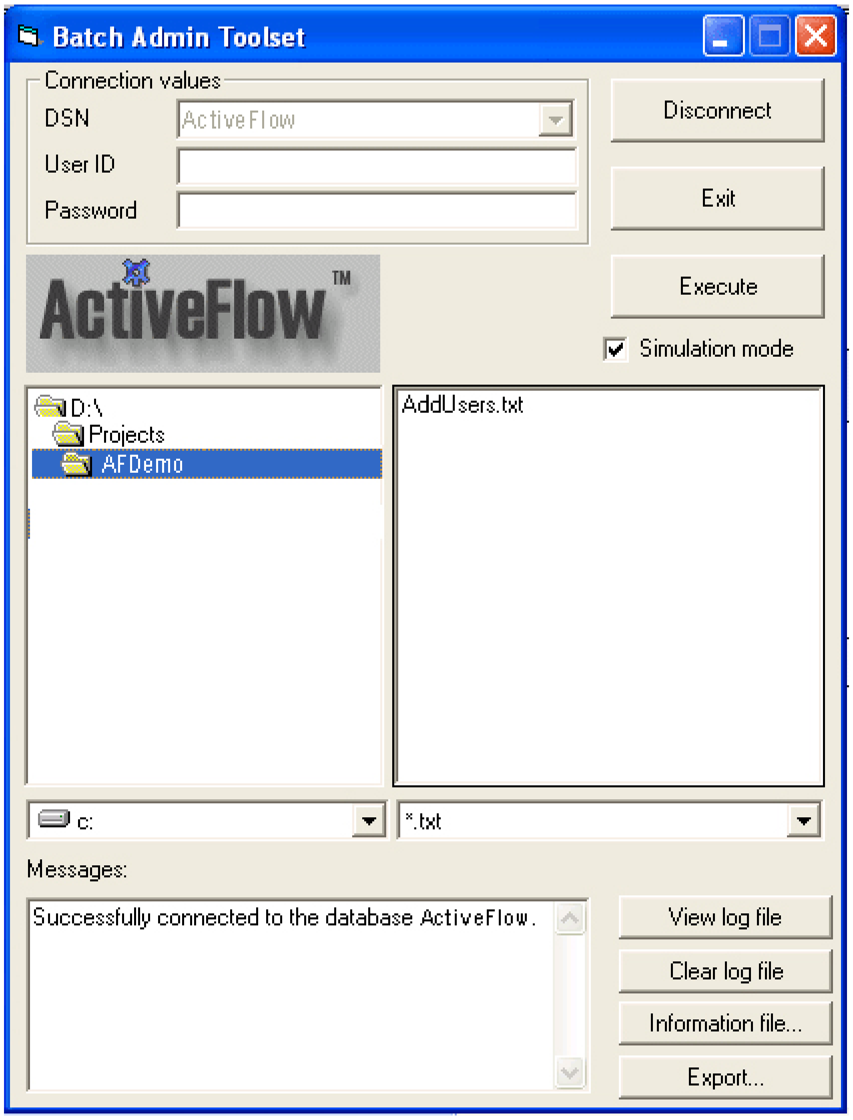
Note:

The organization codes (Role code or Department code) are needed to make use of the Batch Admin Toolset for all these operations.

Using the Batch Admin Toolset

The Batch Admin Toolset (BAT) is a utility program that updates the ActiveModeler/ActiveFlow database according to data in an input file. The input file is basically a text file where changes to the database are specified following strict formatting rules for each of the 12 administrative operations. (See Formatting BAT input files)

The BAT has a very simple and intuitive user interface:



Successfully connected to the database ActiveFlow.

These are the steps to be followed when using BAT to update a database:

1. Connect to the database
2. Select a valid input file
3. Test the input file using Simulation mode
4. Update the database
5. Disconnect from the database

1. Connect to the database

The Batch Admin Toolset must first connect to the ActiveFlow admin database. The administrator makes that connection as follows:

1. Start the ActiveFlow Batch Admin Toolset

2. Complete the required fields:
Make sure the database has been registered. If not go to the Windows Control Panel and select ODBC Data Sources and follow the instructions to register the database.
Make sure that the username is valid, It has to be a person who has the right to update the databases directly. The username and the password do not correspond to an ActiveFlow user; they are related to the database.

3. Press the **Connect** button to connect to the database, or press the **Cancel** button to exit without connecting. If successfully connected, the caption of the **Connect** button changes to "Disconnect" and a confirmation message is displayed in the Message area.

2. Select a valid input file

The input file could be prepared manually each time or, more usefully, it could be automatically created from another system, such as the Personnel system. The file must be a text file with any extension (not only .txt, although we used this extension in the examples) and it should contain only correctly formatted information. The input file can contain one or more administrative operations. The operations executed when the file is processed are specified by operation tags. For detail on this topic check Formatting BAT input files.

When a file is selected by clicking on it in the file control, the program automatically verifies the existence of valid BAT operation tags in that file. The result of this verification is displayed instantly in the Message area. If the file doesn't contain valid operation tags the message "No operation tags found. Please select another file" is displayed. If the file is a valid file the program displays a list of valid operations found in the file:

Example:

```
Operations found in file:
 Load departments
 Load roles
 Load users
```

3. Test the input file using the Simulation mode

Before processing a file in order to change the database it is recommended to process the file in Simulation mode. Enter simulation mode by checking the appropriate checkbox on the BAT panel and start simulation by pressing the **Execute file** button. This way, the input file is processed as in normal mode but no changes are made to the database. All the errors are listed in the report_log.txt file. Also, in case of errors the input file is copied to another file where each line containing an error is marked with the description of the error. For details please see Errors reporting and re-execution files. Correct the errors and then pass to the next step "Update the database".

4. Update the database

To update the database, uncheck the **Simulation mode** check box then press **Process file** button. This action must be used with care as it changes the database.

Any number of files can be processed in a BAT session. You can create separate files for each administrative operation or you can put more operations in the same input file. For each file processed, a report is created in the report_log.txt and also a re-execution file. For details please see Errors reporting and re-execution files.

All the changes to the database are recorded in the AF_Event table and can be viewed by the ActiveFlow superadmin using the View Events form.

5. Disconnect from the database

The Batch Admin Toolset must be disconnected from the ActiveFlow admin database after completing any batch operations. You can disconnect by pressing the **Disconnect** button on the BAT panel or simply closing the BAT program.

FORMATTING BAT INPUT FILES

The Batch Admin Toolset uses a special text file containing the formatted information. The default file extension is .txt, but this can be different as long as the information is formatted as stated below.

A valid file always begins with a special tag on the first line. The tag must be one of the following:

TAG	Operation
<ACTIVEFLOW LOAD USERS>	Load users
<ACTIVEFLOW DELETE USERS>	Delete users
<ACTIVEFLOW MODIFY USERS>	Modify users
<ACTIVEFLOW MOVE USERS>	Move users
<ACTIVEFLOW LOAD DEPARTMENTS>	Load departments
<ACTIVEFLOW DELETE DEPARTMENTS>	Delete departments
<ACTIVEFLOW RENAME DEPARTMENTS>	Rename departments
<ACTIVEFLOW MOVE DEPARTMENTS>	Move departments
<ACTIVEFLOW LOAD ROLES>	Load roles
<ACTIVEFLOW DELETE ROLES>	Delete roles
<ACTIVEFLOW RENAME ROLES>	Rename roles
<ACTIVEFLOW MOVE ROLES>	Move roles

Do not modify the tags in any way, otherwise the BAT cannot recognize the file.

The file must contain the tag and the information related to the operation specified by the tag.

The operation information must be formatted for each operation as stated in the Operations paragraph. There are some general rules that apply to all information in a BAT input file as below:

IMPORTANT:

· Each information line must begin on a new line

· The file must be terminated with at least a new line character (<Enter>)

· Comment lines are allowed and always begin with the special character *

· No other type of comment is allowed

· Blank lines are allowed, they do not affect the processing

· No other type of information is allowed.

More than one operation tag can be present in a BAT input file. The info below each tag is used as input data for the operation specified by the respective tag. For example:

```
<ACTIVEFLOW LOAD DEPARTMENTS>

* A comment line
* Sales department has the root (Company) as parent
'Sales', '101', '0'
'Marketing','103', '0'
'Accounting','107', '0'
'Regional Sales ', '106', '101'

<ACTIVEFLOW LOAD ROLES>

'Sales manager', 'R1012','101'
'Sales representative', 'R1011','101'

<ACTIVEFLOW MOVE DEPARTMENTS>
'106', '0'
```

In the example above, the input file contains more than one tag. The operations will be executed sequentially: first the departments will be loaded, then roles and at the end the department with code 106 will be moved to the department with code 0 (root). The order in the file is important. It is quite obvious that sub departments or roles cannot be added before their parent departments and so on.

DETAILED FORMATTING INSTRUCTIONS

1. Load departments

The **"Load departments"** function reduces the time needed for specifying a new hierarchy of departments for the company. This can be done by adding departments directly from a file.

The tag for this operation is "<ACTIVEFLOW LOAD DEPARTMENTS>". Do not modify the tag in any way, otherwise BAT cannot recognize the file.

Each **department** definition must be formatted as in the table below.

	Data element names	Whether required, and initial setting	Data type	Explanation
1	New Dept Name	required	text	The name of the new department
2	New Dept Code	required	text	The code of the new department. The department code is an alphanumeric string (< 64 char) that identifies a department uniquely in the company)
3	Parent Dept Code	required	text	The code of the parent department. The parent department must exist. If this has the value "0" then the new department is added into the company structure root.

Other rules:

- The Department Root (such as Company) is always marked in the file as '0'

- Parent departments must be added before adding child departments - i.e. in the input file, the child definition must be placed after the parent definition

Example:

A valid file containing four new departments to be loaded:

<ACTIVEFLOW LOAD DEPARTMENTS>
* A comment line
* These three departments have the root (Company) as parent

'Accounting', 'CD101', '0'
'Human Resources', 'CD102', '0'
'Sales', 'CD104', '0'
*'Regional Sales Management' department will have as a *parent the department with code CD104, that is 'Sales'.

```
'Regional Sales Management', 'CD106', 'CD104'
```

Notes:

- <CR> represents a Carriage Return (the Enter key)
- Departments must have a Code assigned in order to be used with "Load departments" tool (See Add a new department). The department code is available in ActiveModeler->Inspecting project...->Structure->(select department)->Settings.

2. Delete departments

This function is used to delete an empty department. Trying to delete a non empty department generates an error. The tag for this operation is "<ACTIVEFLOW DELETE DEPARTMENTS>". Do not modify the tag in any way, otherwise the BAT cannot recognize the file.

Line format:

	Data element names	Whether required, and initial setting	Data type	Explanation
1	Department Code	required	text	Code of the department to be deleted
2	Ghost/ Permanent delete flag	required	number	This flag specifies whether the department will be made ghost or will be deleted completely from the database. Depending on the value of this flag the dept will be: ■ ghosted - if this flag is 0 ■ deleted permanently- if this flag is 1

Note: It is not recommended to delete structure items permanently (physically). This way, no trace of the item remains in the database for future reporting.

Example:

<ACTIVEFLOW DELETE DEPARTMENTS>
'D101', 0
'D102', 1

Note: in the example above, department D101 is ghosted and D102 is deleted permanently from the database.

3. Rename departments

The tag for this operation is "<ACTIVEFLOW RENAME DEPARTMENTS>".

Line format:

	Data element names	Whether required, and initial setting	Data type	Explanation
1	New department name	required	text	The new name of the department
2	Department code	required	text	Code of the department to be renamed

Example:

```
<ACTIVEFLOW RENAME DEPARTMENTS>
'NewName', 'D103'
```

Note: in the example above, dept. D103 is renamed to "NewName"

4. Move departments

Used for moving departments to another place in the organization structure. When moving a department the whole structure below it (depts/roles) is moved. The tag for this operation is "<ACTIVEFLOW MOVE DEPARTMENTS>".

Line format:

	Data element names	Whether required, and initial setting	Data type	Explanation
1	Department code	required	text	Code of the moved department
2	New destination parent department code	required	text	Code of the new parent department

Example:

<ACTIVEFLOW MOVE DEPARTMENTS>
'D106', 'D101'

Note: in the example above, a department with code D106 is moved to department having code D101.

5. Load roles

The **"Load roles"** function is used for adding roles directly from a file into the company structure.

The tag for this operation is "<ACTIVEFLOW LOAD ROLES>".

Each **role** definition must be formatted as in the table below.

	Data element names	Whether required, and initial setting	Data type	Explanation
1	New Role Name	required	text	The name of the new role
2	New Role Code	required	text	The code of the new role. The role code is an alphanumeric string (< 64 char) that identifies a role uniquely in the company)

3	Parent Dept Code	required	text	The code of the parent department. The parent department must exist. If this has the value "0" then the new role is added into the company structure root.

Other rules:

The Department Root (such as Company) is always marked in the file as '0

Example:

A valid file containing four new roles to be loaded:

```
<ACTIVEFLOW LOAD ROLES>
* A comment line

* This two roles has the root (Company) as parent

'System administrator', 'CR1001', '0'
'General manager', 'CR1002', '0'

*Roles 'Sales Manager' and 'Sales representative' will have as a parent the department with code CD104, that is 'Sales' (see examples for Load departments)

'Sales Manager', 'CR1004', 'CD104'
'Sales Representative', 'CR1005', 'CD104'
```

Notes:

- Roles must have a Code assigned in order to be used with "Load roles" tool. (See Add a new role). The role code is available in Properties Browser from ActiveModeler Avantage if the role is selected..

6. Delete roles

This function is used to delete an empty role. Trying to delete a non empty role generates an error. The tag for this operation is "<ACTIVEFLOW DELETE ROLES>". Do not modify the tag in any way, otherwise the BAT cannot recognize the file.

Line format:

	Data element names	Whether required, and initial setting	Data type	Explanation
1	Role Code	required	text	Code of the role to be deleted. the role is not deleted permanently from the database but only ghosted.

Example:

<ACTIVEFLOW DELETE ROLES>
'R1016'

Note: in the example above, a role with code R1016 is ghosted.

7. Rename roles

The tag for this operation is "<ACTIVEFLOW RENAME ROLES>".

Line format:

	Data element names	Whether required, and initial setting	Data type	Explanation
1	New role name	required	text	The new name of the role
2	role code	required	text	Code of the role to be renamed

Example:

<ACTIVEFLOW RENAME ROLES>
'NewRoleName', 'R1013'

Note: in the example above, a role having code R1013 is renamed to "NewRoleName".

8. Move roles

This function is used for moving roles under different departments. When moving a role the users below it are also moved with the role. The tag for this operation is "<ACTIVEFLOW MOVE ROLES>".

Line format:

	Data element names	Whether required, and initial setting	Data type	Explanation
1	Role code	required	text	Code of the moved role
2	New destination parent department code	required	text	Code of the new parent department

Example:

<ACTIVEFLOW MOVE ROLES>
'R1016', 'D101'

Note: in the example above, a role with code R1016 is moved to a department having the code D101.

9. Load users

This Batch Admin Toolset function helps you to add users to the database quickly.Such an automated interface can relieve repetitive work using the online administration forms (especially when many additions are required). You can add any number of users to the database from a formatted file.

The file could be prepared manually each time or, more usefully, it could be automatically created from another system, such as the **Personnel** system.

The header is "<ACTIVEFLOW LOAD USERS>". Do not modify the header in any way, otherwise the **Load users** function cannot recognize the file.

Each **user** definition must be formatted as in the table below:

	Data element names	Whether required, and initial setting	Data type	Explanation
1	User ID	Required	Text	User ID, to be used as a unique identifier. Must not exceed 16 characters.
2	Password	Required	Text	User initial Password. Must not exceed 8 characters.
3	FirstName	Required	Text	User's first name
4	LastName	Required	Text	User's last name
5	Role code	Required	Text	A role with this code must exist in the organization structure otherwise an error occurs.
6	Title		Text	The user's title
7	E-mail		Text	User Email address
8	Phone		Text	User telephone
9	Active	Required (0 or 1)	Number	Shows whether the user is active (1) or not (0)
10	Flags	Required	Number	Represents the user's rights. (0- ordinary user rights, 2 -administrator rights, 4 - superadministrator rights)
11	Hierarchy	Required	Number	User's hierarchy level. Must be a nonnegative number.(0 = the highest level - most senior)

12	Delegate1		Text	User ID of the user's delegate. If provided then it must not exceed 16 characters.
13	DlgActive	Required (0 or 1)	Number	Shows whether the delegate of the user is active (1) or not (0)
14	DelegateMaker		Text	User ID of the user's delegate maker. If provided then it must not exceed 16 characters.
15	DlgMkrActive	Required (0 or 1)	Number	Shows whether the delegate maker of the user is active (1) or not (0)
16	Primary Route		Text	Normal route in the event of bubble-up routing. Since it represents a UserID it must not exceed 16 characters.
17	Alternative Route		Text	Alternative route in the event of bubble-up routing. Since it represents a UserID it must not exceed 16 characters.
18	User custom fields (optional)		Text	Up to 10 custom fields values separated by<,>. The values will be put in the fields User1..10 of the AF_Users table. The number of values can be variable (from 0 to 10).

Make sure that:

· Each **Load user** record begins on a new line

· All the fields are separated by commas

· All Required fields are completed.

Example:

A valid file containing two users to be added to the database:

```
<ACTIVEFLOW LOAD USERS>
*this is a comment line
'user1','demo','Dan','Purcell','CR1005','','dan@email.com', '12344565',1,
2,2,'',0,'',0,'','','000234','dan.p@home-email.com'
'user2','demo','John','Brown','CR1005','','john@email.com', '12344565',1,
2,2,'',0,'',0,'user1','','000235'
```

Notes:

- All the fields must be present, even if they do not contain information. (In the above file, user Dan Purcell doesn't have title, so his title field (immediately after role code -'CR1005') is marked by commas with a space between them.)

- For the first user the last 2 items separated by <,> are user custom fields. In this example they represent the employee ID and home email address. They will be put in the fields User1 and User2 in this order. You will notice that for the second user (user2) home email address field is missing so it will be ignored.

10. Move users

This Batch Admin Toolset function helps you to quickly move users in the company.The following operations can be achieved with this tool:

- move users from one position to another
- add users to a new position (role) without removing from the old position (role)
- delete users from one position without adding to a new position

The header is "<ACTIVEFLOW MOVE USERS>". Do not modify the header in any way, otherwise the **Move users** function cannot recognize the file.

Each **user** definition must be formatted as in the table below:

	Data element names	Whether required, and initial setting	Data type	Explanation
1	User ID	required	text	User ID of the user whose position is changed
2	Old Role Code		text	The code of the old role. The user will be removed from this position (role). If this parameter is missing the user will be not deleted from the old position but only added to the new position specified by New Role Code.
3	New Role Code		text	The code of the new role. The user will be registered to this new position. If this parameter is missing the user will be deleted only from the old position specified by Old Role Code.

IMPORTANT! Old Role Code and New Role Code cannot be both missing.

Example:

In the example below:

- *user1* is removed from role CR001 and moved to role CR002

- *user2* is only removed from role CR003

- *user3* is only registered to the new role CR004 without being removed from the old position (copy)

```
<ACTIVEFLOW MOVE USERS> <CR>
*this is a comment line <CR>
<CR>
'user1','CR001','CR002' <CR>
'user2','CR003',        <CR>
'user3',       ,'CR004' <CR>
```

Notes:

·<CR> represents a Carriage Return (the Enter key)

11. Delete users

This Batch Admin Toolset function helps you to quickly delete or inactivate users in the company.

The header is "<ACTIVEFLOW DELETE USERS>". Do not modify the header in any way, otherwise the **Delete users** function cannot recognize the file.

Line format:

	Data element names	Whether required, and initial setting	Data type	Explanation
1	User ID	required	text	User ID of the user to be deleted
2	Inactivate/ Delete flag	required	number	This flag specifies whether the user will be inactivated or will be deleted completely from the database. Depending on the value of this flag the user will be: ▪ inactivated - if this flag is 0 ▪ deleted -if this flag is 1

Note :

A user involved in some ActiveFlow transactions cannot be deleted completely from the database. In this case an error will be reported.

Example:

In the example below:

- *user1* is inactivated (Active property of the user will be set to 0)
- *user2* is completely removed from the database

```
<ACTIVEFLOW DELETE USERS>
*this is a comment line
<CR>
'user1',0
'user2',1
```

12. Modify users

This Batch Admin Toolset function helps you to quickly modify properties of users in the company.

The header is "ACTIVEFLOW MODIFY USERS".

Each line must be formatted as in the table below:

	Data element names	Whether required, and initial setting	Data type	Explanation
1	User ID	Required	Text	User ID, to be used as a unique identifier. Must not exceed 16 characters.
2	Password		Text	User's password. Must not exceed 8 characters.
3	FirstName		Text	User's first name
4	LastName		Text	User's last name
5	Title		Text	The user's title
6	E-mail		Text	User Email address
7	Phone		Text	User telephone
8	Flags		Number	Represents the user's rights. (0- ordinary user rights, 2 -administrator rights, 4 - superadministrator rights)
9	Hierarchy		Number	User's hierarchy level. Must be a nonnegative number.(0 = the highest level - most senior)
10	Delegate		Text	User ID of the user's delegate. If provided then it must not exceed 16 characters.
11	Delegate active		Number	Shows whether the delegate of the user is active (1) or not (0)
12	DelegateMaker		Text	User ID of the user's delegate maker. If provided then it must not exceed 16 characters.

13	DlgMkrActive		Number	Shows whether the delegate maker of the user is active (1) or not (0)
14	Primary Route		Text	Normal route in the event of bubble-up routing. Since it represents a UserID it must not exceed 16 characters.
15	Alternative Route		Text	Alternative route in the event of bubble-up routing. Since it represents a UserID it must not exceed 16 characters.
16	User custom fields (optional)		Text	Up to 10 custom fields values separated by<,>. The values will be put in the fields User1..10 of the AF_Users table. The number of values can be variable (from 0 to 10).

Note:

None of the above fields is required apart from the UserID. Only the provided fields will be updated.

Example:

In the example below:

- for *user1* will change the password
- for *user2* will change the delegate name and Delegate flag
- for *user3* will change the email address, the phone and also the home email address. In this case you will notice that the User1 field (which was used in the previous example in the load users section for storing the employee id) is empty so it will be ignored.

```
<ACTIVEFLOW MODIFY USERS>
*this is a comment line
'user1','newpsw',,,,,,,,,,,,
'user2',,,,,,,,,,'newdelegate',1,,
'user3',,,,,'new@email.org','1234567',,,,,,,,,,'new-address@home-email.com'
```

Error reporting and re-execution files

The reports file: report_log.txt

For each input file processed with the BAT, a new report is appended to the report_log.txt file. A report contains the date/time when the file was processed, the name of the input file and a summary of the errors found and successfully processed lines.

Here is an example:

```
***************************************************************************
***********
Processed on: 2001/05/02 18:01
Input file : D:\Projects\BatchAdminToolset\chg_org.txt


Operation: Load departments
 Line 8 ('Accounting','107, '...): Invalid data format.
 Line 9 ('Regional Sales ', '...): The department code is already used:106

 Lines processed successfully: 3
 Lines with errors : 2

Operation: Load roles
 Line 13 ('Sales manager', 'R1...): The role code is already used:R1012

 Lines processed successfully: 2
 Lines with errors :1

Operation: Move departments

 Lines processed successfully: 1
 Lines with errors : 0


Total lines processed successfully: 6
Total lines with errors : 3
```

As it can be seen in the above report, a list of errors and a partial summary is generated for each operation found in the input file. (e.g. for the first operation, 3 departments were added to the database and and two lines with errors were found). At the end, a total summary is presented. Only the lines that contain data information are considered when generating reports. Comment lines and blank lines are ignored.

Re-execution files

For each input file processed with the BAT, a so called re-execution file is generated along with the report. The name of the re-execution file is generated from the name of the input file to which the suffix "_re" is added. For example, if the input file was called "changeorg.txt", the re-execution file will be called "changeorg_re.txt".

A re-execution file contains all the lines in the original input file which contained errors along with the corresponding operation tags. Before each line of data, a comment line containing the description of the error is inserted. Thus, correcting the errors become very easy. Just step through the re-execution file and correct the errors. After the errors are corrected the file can be processed as a new input file for the BAT.

In the case of Simulation mode, the re-execution file contains all the lines from the original input file, having a comment line with the description of error in front of all lines containing errors.

GENERATING THE STRUCTURE INFORMATION FILE

The structure information file is a file generated from the organization structure existing in the database. This is only an information file and cannot be directly used as input file to the BAT. The organization structure information is formatted as below:

```
******************************
Departments
******************************
Department: Dept name, dept code - parent department name
Department: child dept name1 , child dept code 1
Department: child dept name2 , child dept code 2
 ..................
 Role: role name 1, role code 1
 Role: role name 2, role code 2
 ..................
.....................
.....................
******************************
Roles
******************************

Role: role name , role code - parent dept name
User: User name, user ID, active/inactive
 .............
 ...............................

******************************
Users
******************************
User: FirstName, LastName, User ID, Title, E-mail, Phone, Active,Flag, Hierarchy, Delegate,
DlgActive, DelegateMaker, DlgMkrActive, PrimaryRoute, AlternativeRoute
 user's role 1
 user's role 2
 ...........
..................................................
```

The **departments section** contains all the department names, codes and parent department names, as well as the first level of child departments and roles for each department.
The **roles section** contains all the roles with name, code, parent department and the users in that role.
The **users section** contains all the users sorted by the last name and the roles each user belongs to.

The steps for generating an information file are:

1.Press the ***Information file...*** button

2. Specify a file where the structure information will be written

3. Press ***Save***

Exporting the organization structure to a BAT execution file

The organization structure existent in an ActiveFlow database can be exported back to a text file in the same format as the BAT input files. This file can be used as input file for populating a new ActiveFlow database.

For exporting structure to a file press **Export...** button and chose the destination file.

Using the BatchAdminToolset from command line

The BatchAdminToolset (BAT) can be used from the command line. For this, please use the AFBAT_cmd.exe executable file instead of AFBAT_win.exe, which is the Win 32 interface version. No message boxes will be displayed. All the messages will be saved into the report_log.txt file.

The syntax for using BAT in command line is:

AFBAT_cmd *filename dbdsn* [*dbusr dbpsw*]

Where:

filename - specifies the full path of the input file

dbdsn - the name of the database(or Data Source Name) as registered in the ODBC

dbusr - the username of a database administrator. (Only if required)

dbpsw - the password for the above username. (Only if required)

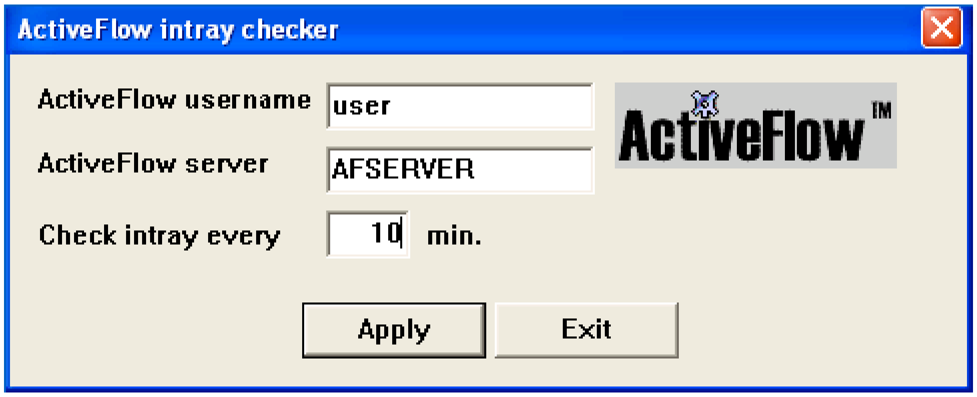
Example:

AFbat_cmd d:\docs\chg_org.txt AFCompanyDB

In-tray notifier

The ***In-tray notifier*** is an independent (and optional) application which periodically checks the user's In-tray and if the number of work items has increased, a notification dialog box appears.

The In-tray notifier application has to be installed on each client computer and the first time it is run, the user will be prompted to enter the ActiveFlow username, ActiveFlow Information provider server and the timer value as below:


ActiveFlow intray checker

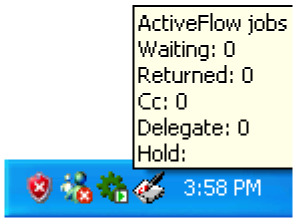
ActiveFlow username: user

ActiveFlow server: AFSERVER

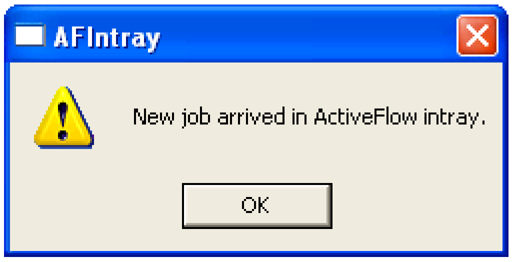
Check intray every 10 min.

When running, the application shows an icon in the task bar

On "mouse-over" of this icon the **In-tray notifier** pop-up is shown as below:

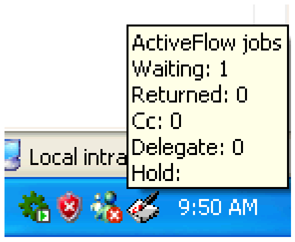

ActiveFlow jobs
Waiting: 0
Returned: 0
Cc: 0
Delegate: 0
Hold:

When a new workflow item arrives for the user the following message appears on the screen:


AFIntray

New job arrived in ActiveFlow intray.

On "mouse-over" of the start bar **In-tray notifier** icon we see it is now updated as below:


ActiveFlow jobs
Waiting: 1
Returned: 0
Cc: 0
Delegate: 0
Hold:

By double-clicking the icon the **ActiveFlow In-tray checker** dialog is displayed as above, allowing the user to change the settings.

Note:

The application will start when the user logs on to Windows.
The In-tray notifier application uses port 1973 in order to communicate to ActiveFlow information provider application so developers should avoid using this port for other applications.

Virtual client

The virtual client consists of set of ActiveX objects(contained in AFInterface.dll) which allows external systems to connect via COM to the ActiveFlow engine in order to enter/extract data from the ActiveFlow database. The interface object resembles the behavior of an internet browser: it connects to the web server and submits requests using HTTP protocol.

The virtual client object can do the following actions:

- Get the list of waiting jobs for a user
- Get the field values for a certain job
- Handle an existing form
- Start a new workflow

Note:

Before doing any of the above actions, the interface object has to be logged on to ActiveFlow.

Initialization

In order to use the ActiveFlow adapter, some properties are required to be initialized before calling its API. These properties are:

Property	Description
AFServer	The name of the ActiveFlow server
AFApplication	The ActiveFlow application name

API

Function	Remarks
Login(username, password)	*As with any other ActiveFlow user, the ActiveFlow virtual client must also log on the ActiveFlow system. For this, a valid ActiveFlow username and password are required.* *The function returns True if the login was successful and False otherwise.*
GetJobsList()	The function returns a Dictionary object containing AFJobItem objects.
GetJobFields(jobItem)	The function takes as input parameter a valid AFJobItem object and returns a dictionary object having as keys the field name and as items the appropriate field values.
AddField(fieldName,fieldValue)	In order to start a new workflow, the designer must set the list of pairs <fieldName,fieldValue>.
ClearFields()	Removes all the pairs <fieldName,fieldValue>.
StartNewWorkflow(activityID)	The activityID must be a valid ID of a starting activity. Before using it, the list of pairs <fieldName,fieldValue> should be initialized.
HandleJob(jobItem)	Submits the form associated with the specified job.

Note:
1) The GetJobFields function overwrites the existing field values with the fields values of the current job.
2) The field names values should match the field names in the form associated to the activity.
3) For returning/rejecting a job, the following field values have to be set:

Action	Field name	Value
Return to maker	_AFRR	_AFReturnM
Return to previous	_AFRR	_AFReturnP
Reject	_AFRR	_AFReject

In this case also the field "_AFRRComments" should be initialized with the comments.

AFJobItem object

The AFJobItem object contains job details.

Property	Type	Remarks
ActivityID	string	The ID of the map activity.
Date	string	The date when the job entered the user's In-tray
FlowID	string	The ID unique identifying a workflow from maker to final approver. .It is internally used by ActiveFlow engine and it shouldn't be modified.
JobID	string	The ID unique identifying a job. It is internally used by ActiveFlow engine and it shouldn't be modified.
JobTitle	string	Contains the value of the _AFFormTitle field (see FormTitle AF Control).
Type	integer	The type of the job. 0 - waiting 1 - returned
Workflow	string	The workflow name.

VIRTUAL CLIENT SAMPLE PROJECT

Rather than being a user guide for the sample application, this section targets application developers who have an understanding of using COM objects. The sample application shows how the VirtualClient object can be used to send/receive data to/from the ActiveFlow system. This application was developed using Visual Basic 6.0.

The ActiveFlow installer installs it in the ActiveFlow\Examples\VirtualClient folder.

SETTINGS:

1. Make sure the paths to files attached to activities and output directory are pointing to the appropriate location.
2. Set an ODBC DSN named "VCSample" using the VCSample.mdb database.
3. Run the workflow wizard and if the application virtual directory cannot be set, create it and name it "VCSample".
3. Make sure AFInterface.dll is registered.
4. Make sure that AFExtensios service is running on the ActiveFlow server and has correct settings.

By default, the sample database has the following users:

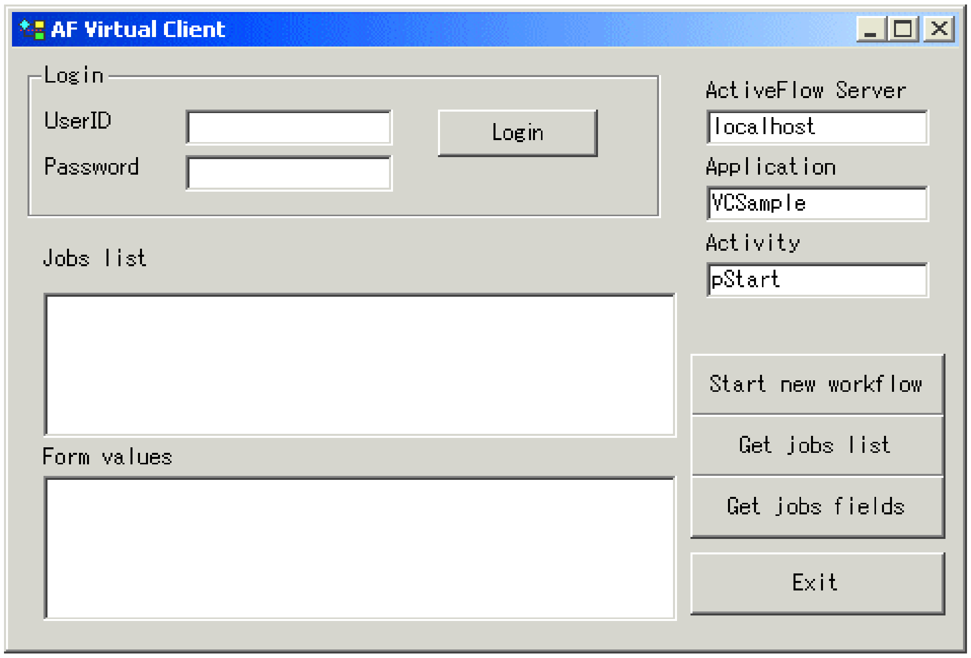
UserID	Password
admin	demo
user1	demo

user2	demo
user3	demo

The candidates for activities are as follows:
user2 for Check
user3 for Approve

User1 issues the form, user2 checks it and user3 finally approves.

The VB directory contains the Visual Basic sample project.



Global variables:
vc - the VirtualClient interface object.
oJobsList - a dictionary object containing the list of jobs (AFJobItem objects) for the user currently logged on.

Functions:
Login - logs on to ActiveFlow system
Start new workflow - will create a new workflow having the following field values:
_AFFormTitle = "Form #<crtNo>"
FieldN = <crtNo>
FieldS = "random number <random number>"
crtNo is the currently submitted form.
Get jobs list - will display in the Jobs list listbox the details (workflow - form title - date) of the waiting jobs of the user currently logged on.
Get jobs fields - will display for the currently selected job, the pairs <field name = field value>

ActiveFlow Tutorial Set

This tutorial set has been developed for staff who are new to ActiveFlow.
The purpose is to show you how to create some common workflow types so you can quickly get up to speed with ActiveFlow.
We assume here a basic knowledge of ActiveModeler and before creating any workflows you must first create an ActiveModeler project to act as a repository of automated business processes - this is explained in Item 1 below.

Let's now have a look at the content of the Tutorial set:

1. ActiveModeler Project Tutorial	This is a step-by-step guide which shows you how to: - Create the ActiveModeler/ActiveFlow database - Create the organization structure - Add the ActiveFlow users - Set the workflow server properties
2. Holiday Request workflow	This workflow presents bubble-up routing. **Business process description:** An employee fills a holiday request form and submits it. It flows up to the department manager level for approval. The form then passes to the Human Resources department for handling.

1. ActiveModeler project tutorial

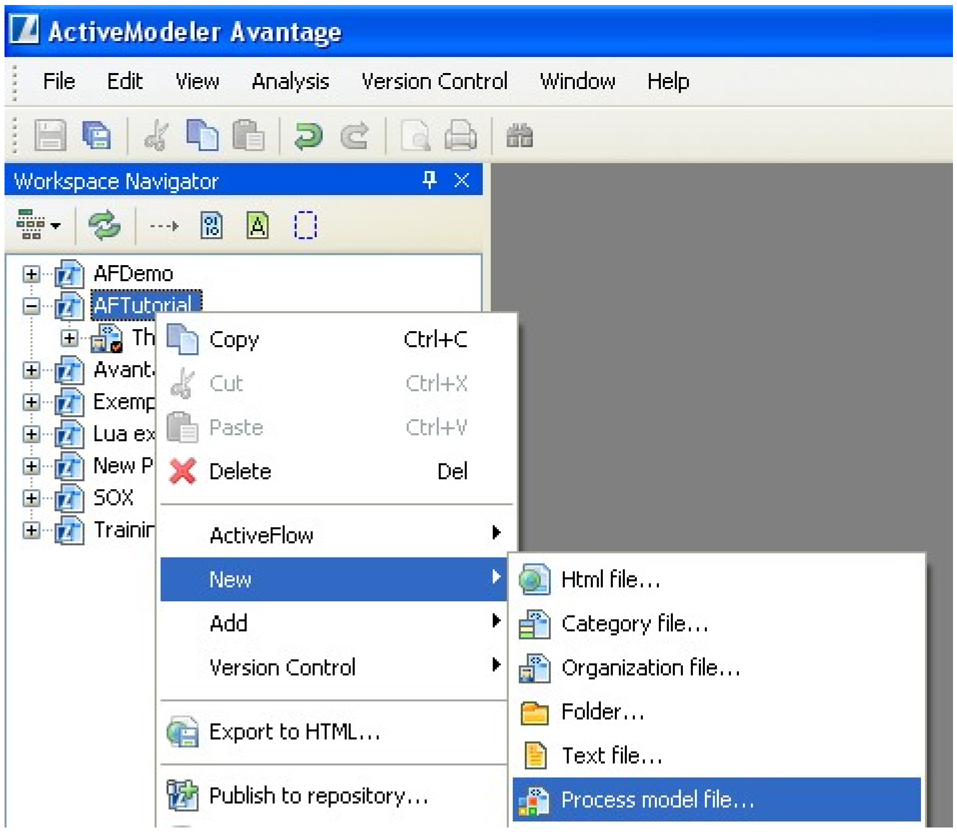
Here we describe the steps to create the workflow project. This is repository of required information and has to be created only once. It will contain global information about the organization structure, ActiveFlow server settings and the workflow maps themselves. After creating the project, the designer can proceed to creating the actual workflows.

Step 1 - Create the project file

Select **File** : "**New project …"** to create a new project
Specify the project name. In our example here, we choose the name "**AFTutorial**" and the path D:\Projects\. You can choose another path.

Then right click on the project item and select **New..->Process model file.** Specify the name for this.

Note:

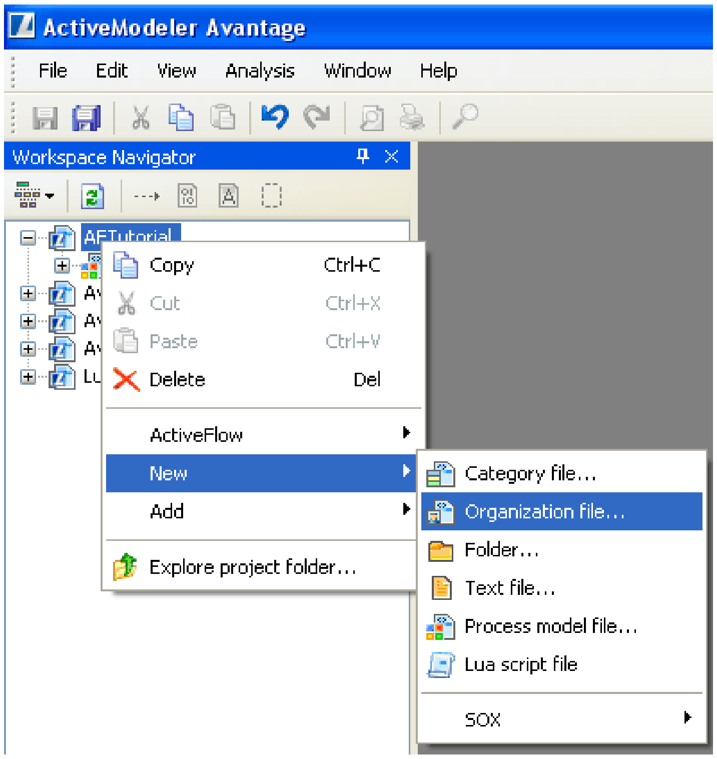


The project contains 3 folders: AsIs, ToBe and Automated. Please consult the ActiveModeler documentation for more details regarding their usage. We will be putting the workflows in the Automated folder.

Step 2 - Create the organization structure

When the project has been created it is necessary to define the organization structure, i.e. the departments, roles and the users.

Creating a new organization file for the project



1. Right click on the project item and select **New->Organization file**
2. Call it **Organization** for now
3. Then right click on the new organization item, select **New..** and add organization items like departments and roles.

Use the **Add Dept...** and **Add Role...** buttons from the right side of the screen to add departments and roles. Under a department you can define another department(s) and/or roles. For each department and role insert the code using the Properties Browser.
Create the structure shown in the dialog above by adding the following departments and roles:

Department name	Code
Any	CD-ANY
Accounting	CD-ACC
IT	CD-IT
Human Resources	CD-HR
Sales	CD-SALES

Role name	Department	Code
Any	Any	CR-ANY

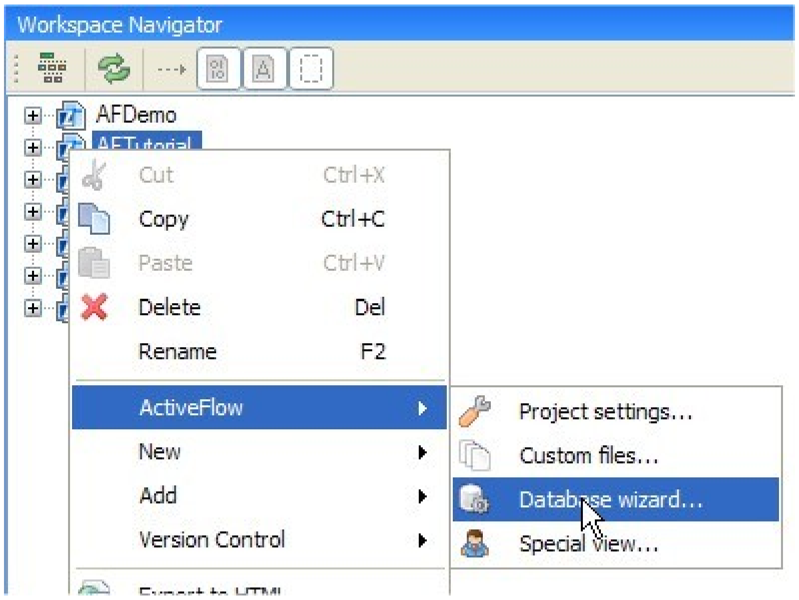
Robot	Any	CR-ROBOT
Clerk	Accounting	CR-CLK-ACC
Manager	Accounting	CR-MGR-ACC
Engineer	IT	CR-ENG-IT
Manager	IT	CR-MGR-IT
Clerk	Human Resources	CR-CLK-HR
Manager	Human Resources	CR-MGR-HR
Clerk	Sales	CR-CLK-SALES
Manager	Sales	CR-MGR-SALES

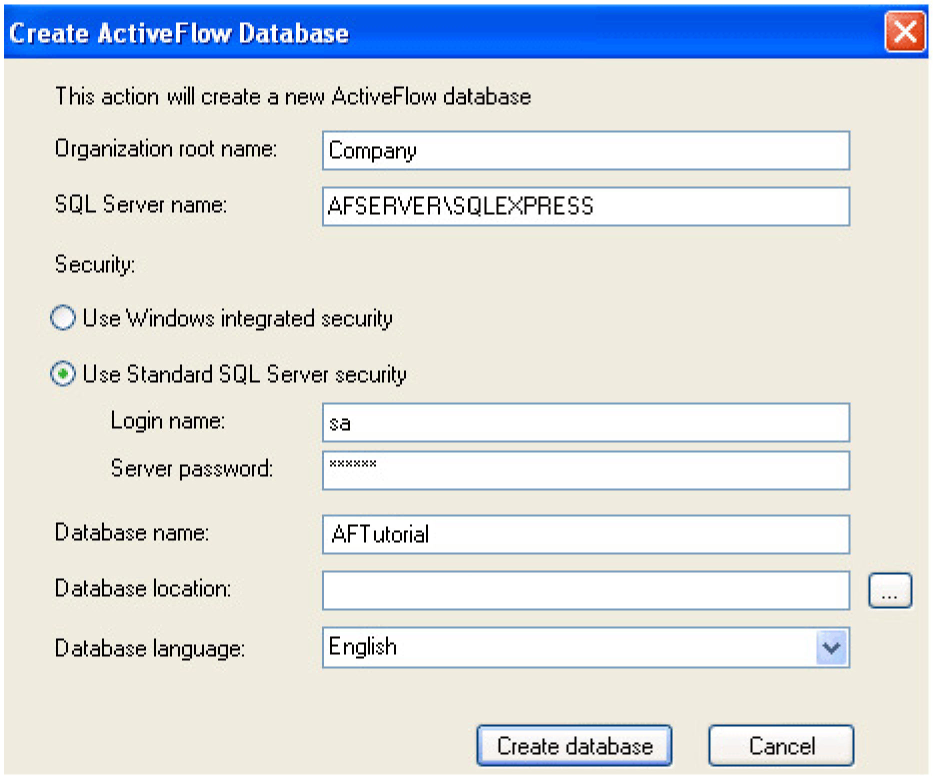
Creating the ActiveFlow database

Right-click on the project item and then select **ActiveFlow -> Database wizard..**

Creating a new database

- Right-click the project item then select **ActiveFlow -> Database wizard..**




Create ActiveFlow Database

This action will create a new ActiveFlow database

Organization root name:	Company
SQL Server name:	AFSERVER\SQLEXPRESS

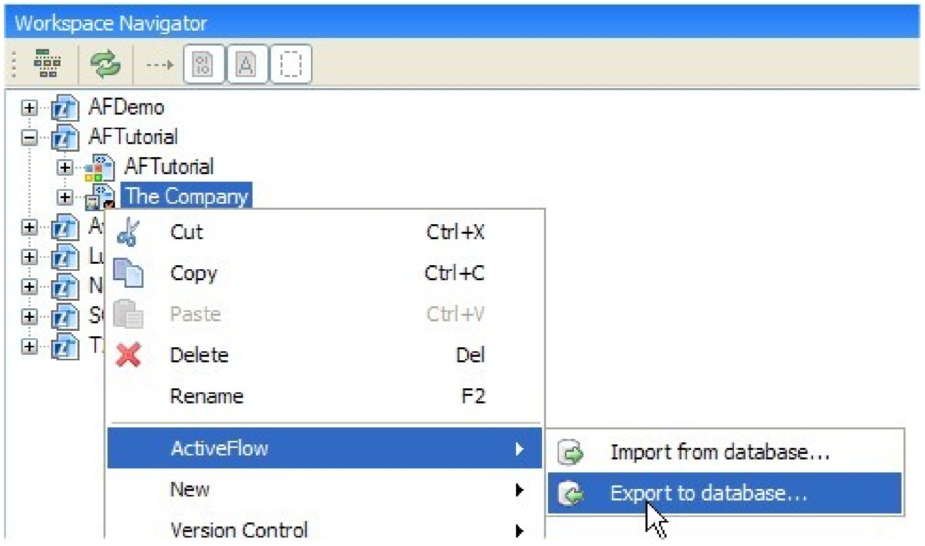
Security:

Use Windows integrated security [✓ Use Standard SQL Server security]

Login name:	sa
Server password:	******
Database name:	AFTutorial
Database location:	
Database language:	English

Notes:
The **Database location** must be the local path where the database files are saved on the SQL server machine. Ask the server administrator for details.

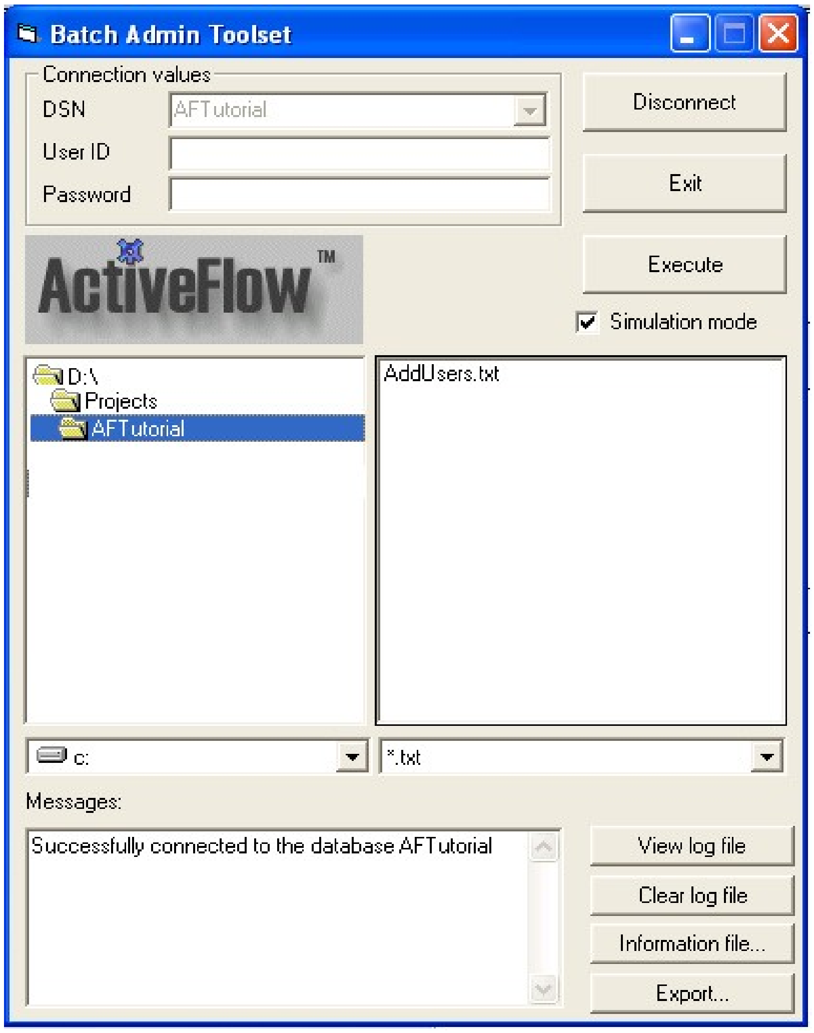
Next we need to export the organization from the modeler to the database using **"Export to database..."** function. Right click on the organization structure to do this as follows:



Step 3 - Add the ActiveFlow users

After creating the above organization structure using ActiveModeler, we will have to add the ActiveFlow users. To do this we will this time use the Batch admin tools. For more details regarding this tool, please consult the Batch admin toolset .
In the *D:\Projects\AFTutorial* directory create a text file *AddUsers.txt* with the following data:

```
<ACTIVEFLOW LOAD USERS>
'paul','demo','Paul','T.','CR-CLK-ACC','Clerk', , ,1,0,80, ,0, ,0,'dan', , ,
'dan','demo','Dan','S.','CR-CLK-ACC','Clerk', , ,1,0,60, ,0, ,0,'smith' , , ,
'smith','demo','Smith','A.','CR-MGR-ACC','Manager', , ,1,0,50, ,0, ,0, , , ,
'marco','demo','Marco','J.','CR-ENG-IT','Engineer', , ,1,0,70, ,0, ,0,'long', , ,
'long','demo','Long','G.','CR-MGR-IT','Manager', , ,1,0,50, ,0, ,0, , , ,
'ray','demo','Ray','S.','CR-CLK-HR','Clerk', , ,1,0,80, ,0, ,0,'almond' , , ,
'scott','demo','Scott','C.','CR-CLK-HR','Clerk', , ,1,0,80, ,0, ,0,'almond', , ,
'almond','demo','Almond','M.','CR-MGR-HR','Manager', , ,1,0,50, ,0, ,0, , , ,
```



DSN: AFTutorial

☑ Simulation mode

D:\
Projects
AFTutorial

AddUsers.txt

c:

*.txt

Messages:

Successfully connected to the database AFTutorial

A complete description of the file format can be found in the Load users section of the Batch admin documentation.

Launch the Batch admin application from the *Start\All Programs\ActiveFlow\Batch admin toolsset* menu.

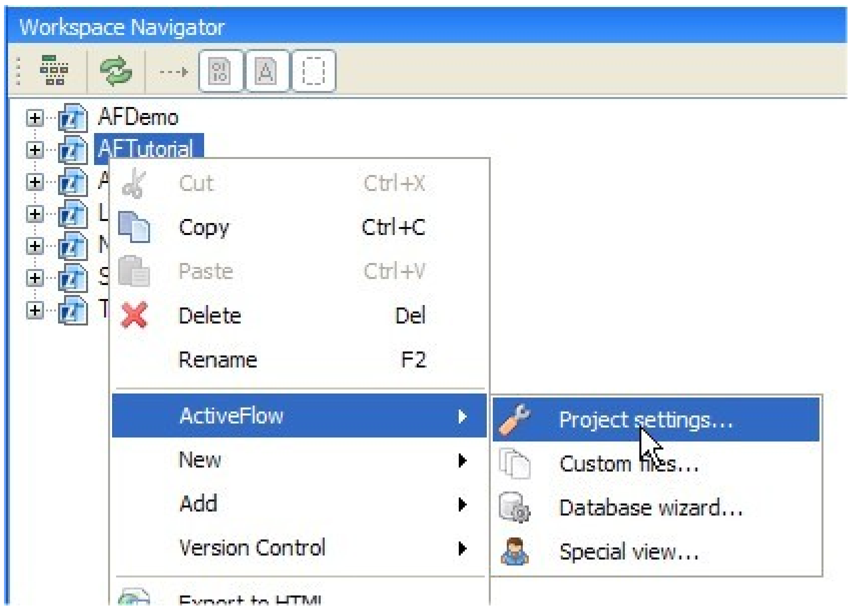
The DSN listbox will display all the available DSNs. Select the **AFTutorial** which is the DSN for the tutorial database and then press the **Connect** button on the right hand side. From the *D:\Projects\AFTutorial* directory select the *AddUsers.txt* file and then press **Execute**. If the operation is successful (no errors were reported in the Messages area) then uncheck the **Simulation mode** checkbox and press the **Execute** button again.

This will create the following organization structure:

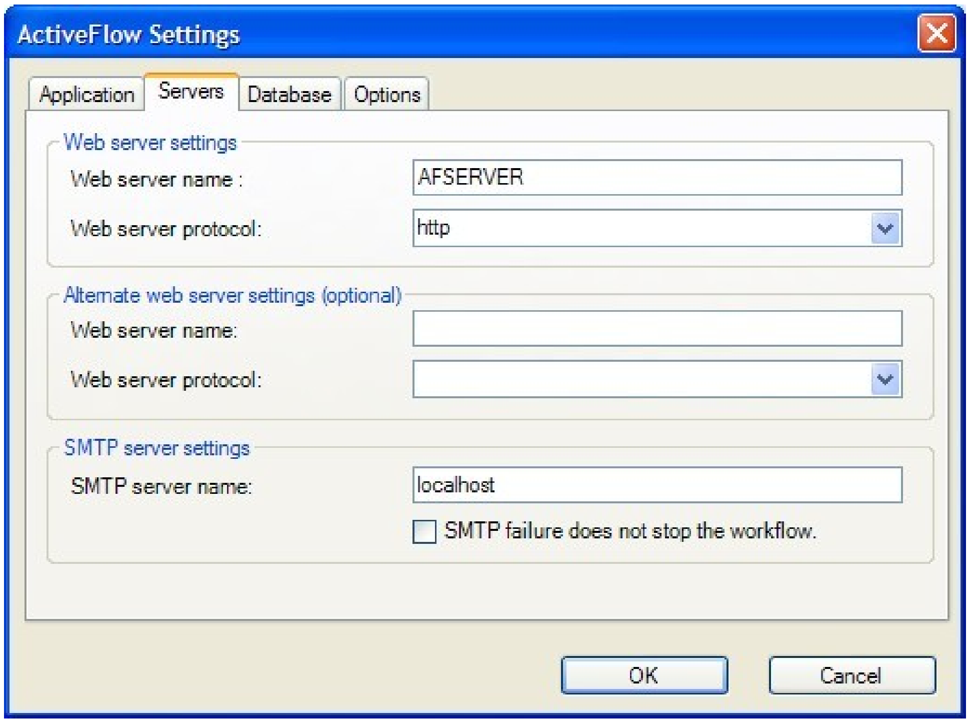
Full Name	User ID	Department	Role
T. Paul	paul	Accounting	Clerk
S. Dan	dan	Accounting	Clerk
A. Smith	smith	Accounting	Manager
J. Marco	marco	IT	Engineer
G. Long	long	IT	Manager
S. Ray	ray	Human Resources	Clerk
C. Scott	scott	Human Resources	Clerk
M. Almond	almond	Human Resources	Manager

Step 4 - Set the server properties

Right-click the project item and select **ActiveFlow**-> **Project** Settings..



- Edit Servers tab - web server name: *AFSERVER*


ActiveFlow Settings

Web server settings

Web server name : AFSERVER

Web server protocol: http

Alternate web server settings (optional)

Web server name:

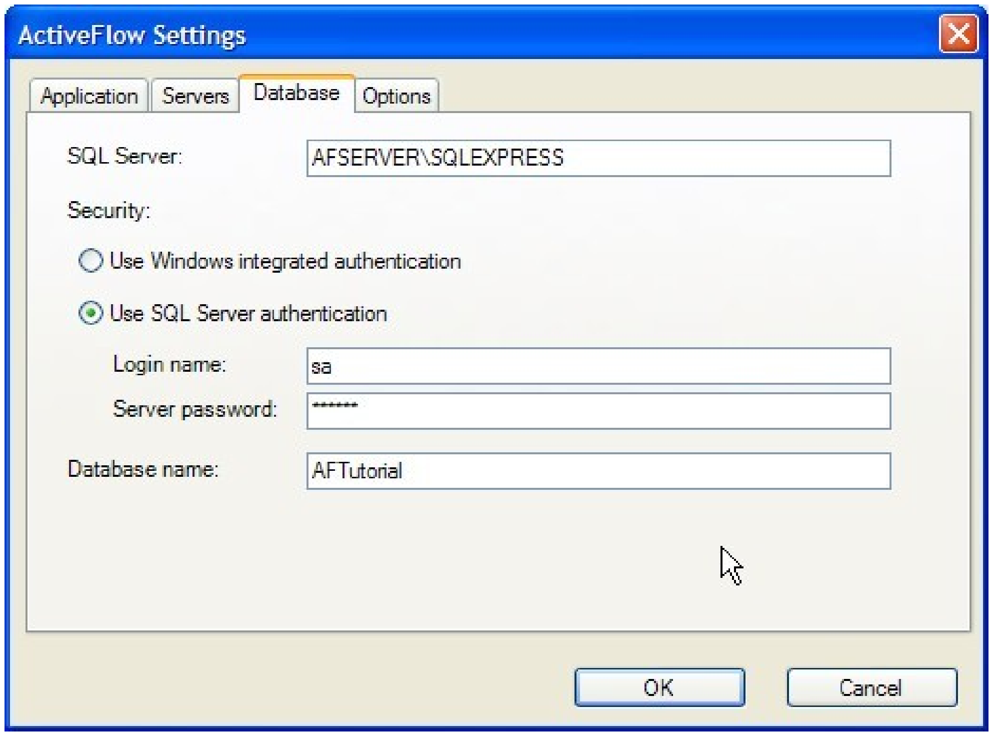
Web server protocol:

SMTP server settings

SMTP server name: localhost

☐ SMTP failure does not stop the workflow.

- Edit Database tab - database name: *AFTutorial*


ActiveFlow Settings

SQL Server: AFSERVER\SQLEXPRESS

Security:

○ Use Windows integrated authentication

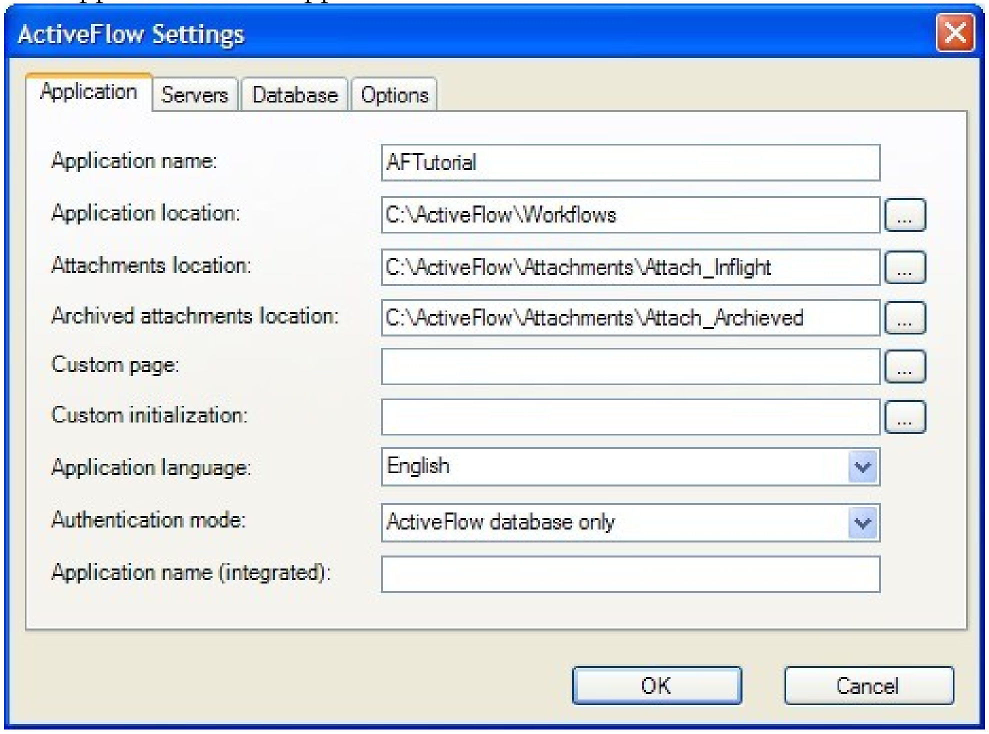
● Use SQL Server authentication

Login name: sa

Server password: *******

Database name: AFTutorial

■ Edit Application tab - application name: *AFTutorial*



Application name: AFTutorial
Application location: C:\ActiveFlow\Workflows
Attachments location: C:\ActiveFlow\Attachments\Attach_Inflight
Archived attachments location: C:\ActiveFlow\Attachments\Attach_Archieved
Custom page:
Custom initialization:
Application language: English
Authentication mode: ActiveFlow database only
Application name (integrated):

Application name	The name of the ActiveFlow web directory on the server
Application location	The network path of the folder on the server where the ActiveFlow files will be generated.
Attachments location	The local path on the ActiveFlow server that is used for attachments of the in progress workflows. Ask the server administrator if you don't know the local path on the server.
Archived attachments location	The local path on the ActiveFlow server that will be used for archiving files attached to archived workflows.
Language	Specify the ActiveFlow default language. The language used for displaying the Login page etc.
Authentication mode	Specify whether to use integrated authentication (ActiveDirectory) or not.

2. ActiveFlow tutorial - Holiday Request workflow

This workflow is one we all like to use! The business logic depends of course on your organization - for this example we have defined it as follows:

- A user fills a form for a holiday request (any user may fill this form).
- The form will "bubble-up" for approval to the Department manager level (the user's title must be "Manager")
- After the Department manager's approval, the form will be sent to the Human Resources department where a clerk will check the form and if everything is fine will finally approve it.

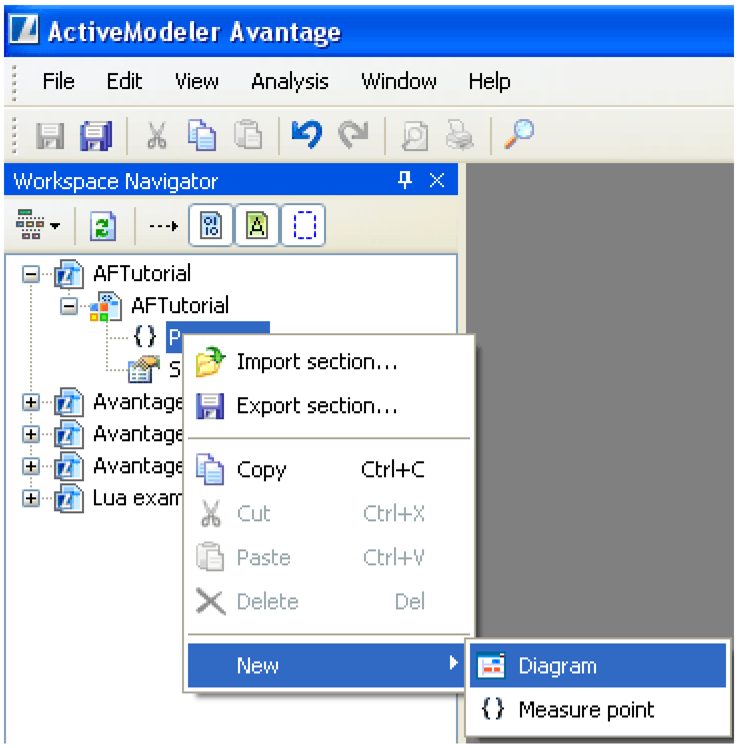
In a real world situation, the designer may want to automatically update an external database (e.g. the Human Resources holiday master file). The tutorial will indicate the places where the designer should write the custom script to achieve the external database access.

Step 1 - Create the workflow diagram

We need to create the workflow diagram and attach it to the project (AFTutorial).

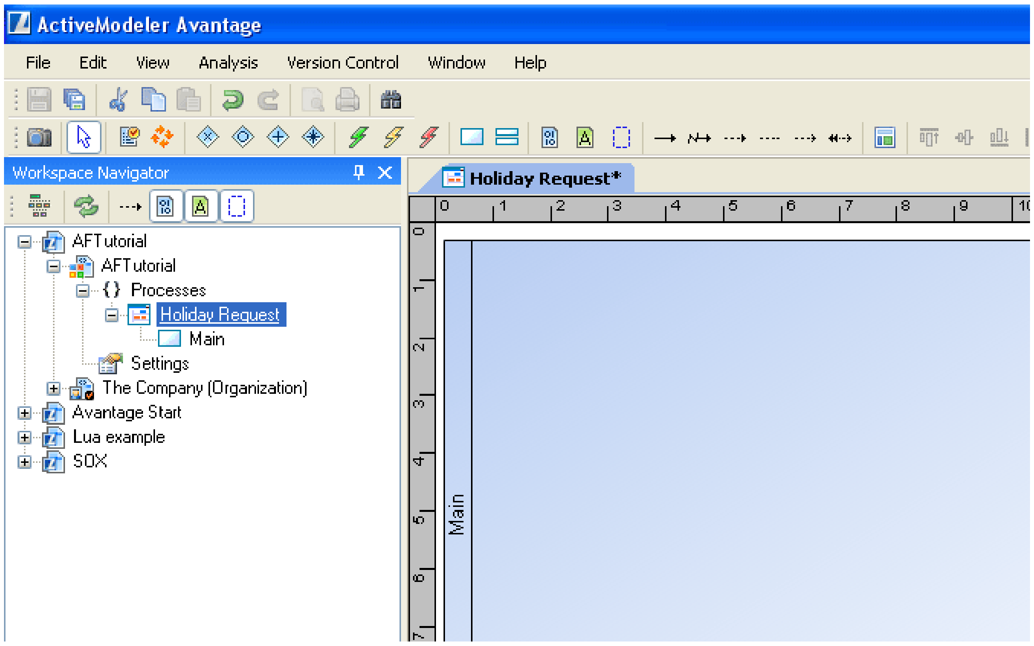
Create a new empty diagram :

Add a diagram in the AFTutorial process model file by selecting Processes: New: Diagram . Call it **"Holiday Request".**



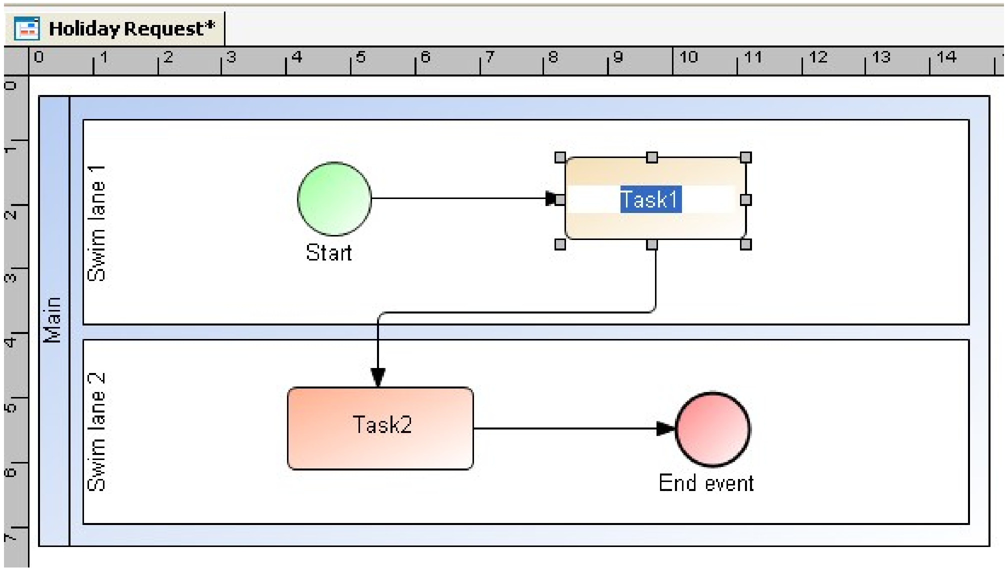
Step 2 - Design the workflow

Open the **Holiday Request** diagram by double clicking on the entry from the Business Process tree (BP tree).

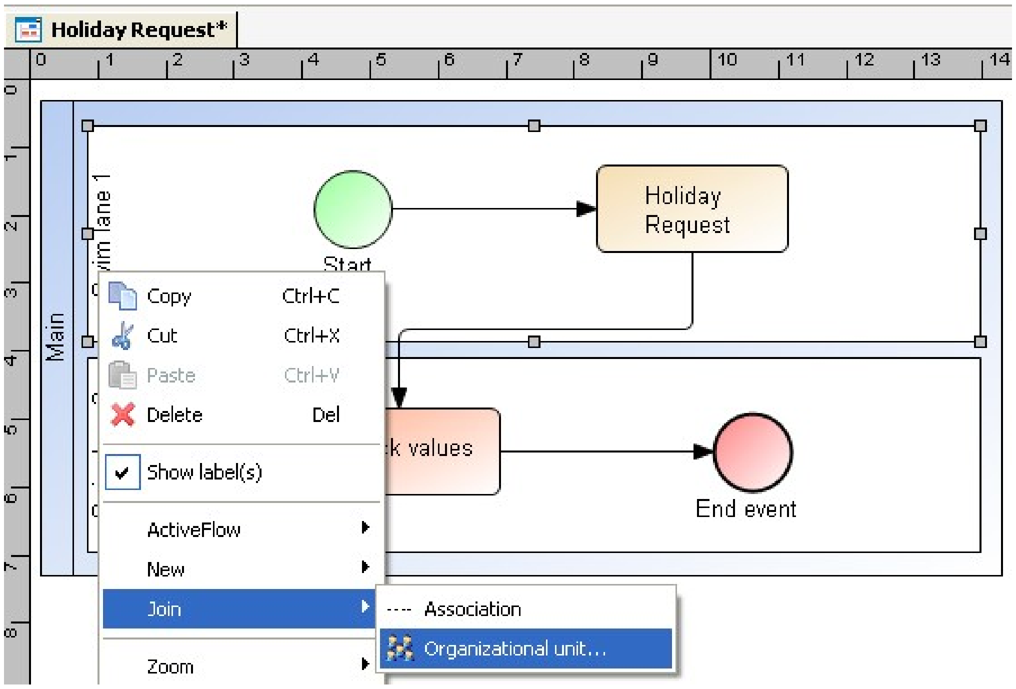


We now need to design the workflow by placing the activities on the diagram in the correct department/role swim lanes, connect them according to the business logic and define the routing rules in the case of splits .

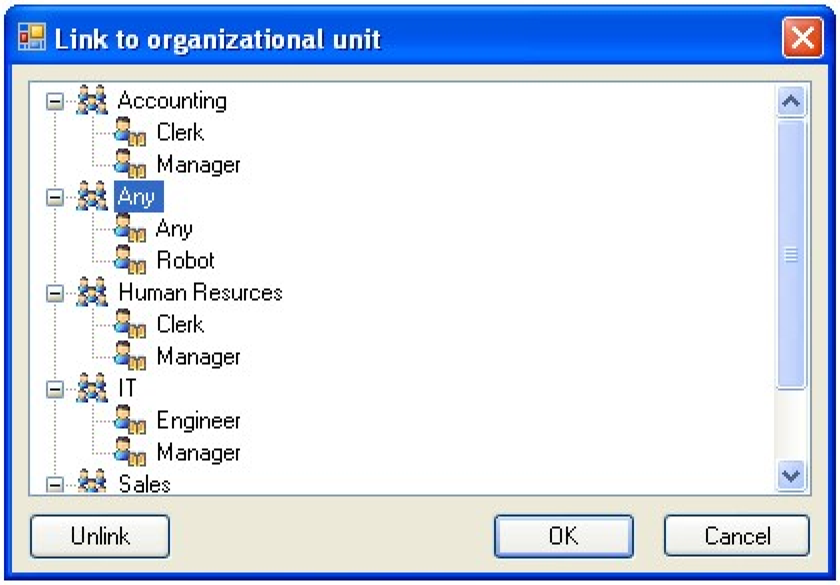
So let's place activities on the diagram. Define the activity caption by double-clicking on the activity shape on the diagram.



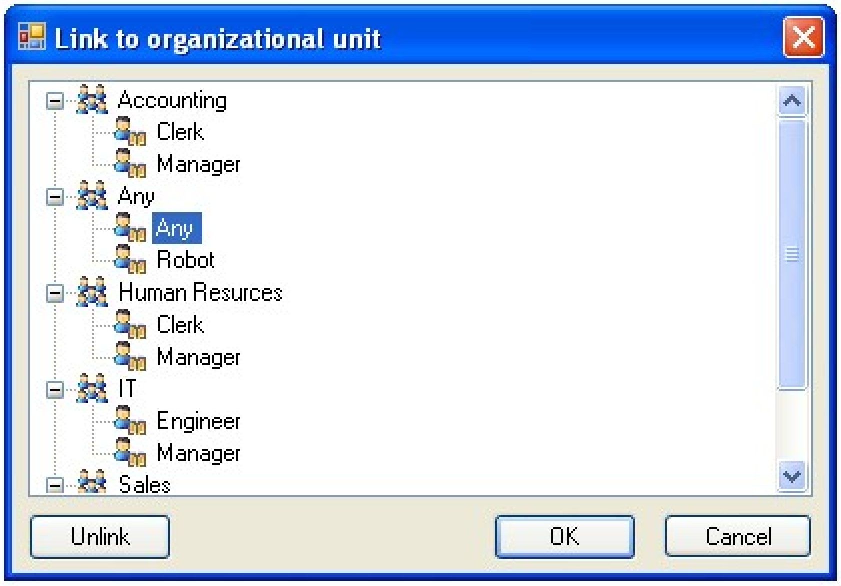
Link the lanes to the appropriate department and role items from organization structure. In order to do this, right-click on the lane and select the **Organizational unit** from **Join** menu.



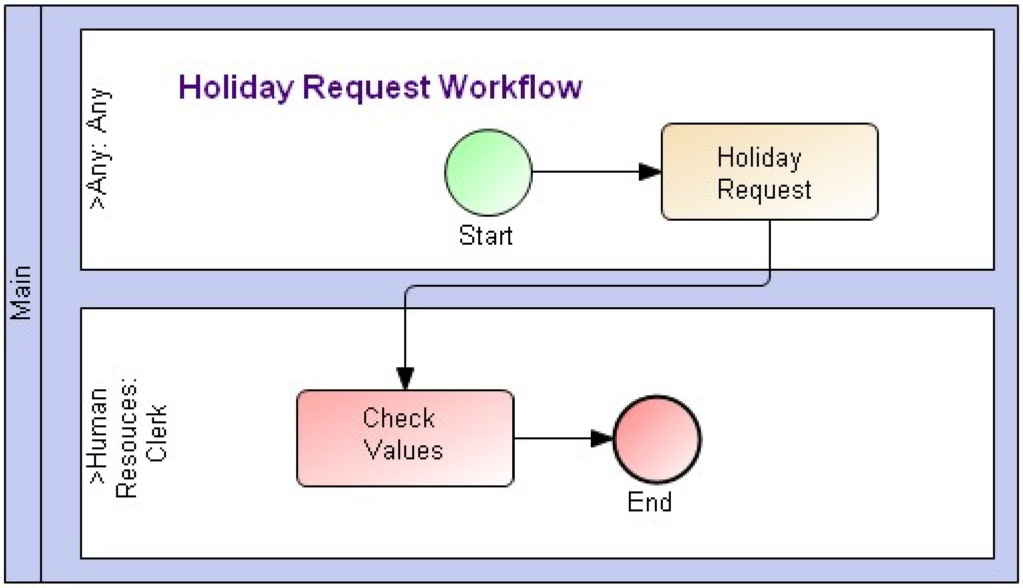
From the organization structure dialogs presented, choose the appropriate department.



Choose the role from the department:



For the Holiday Request example workflow, the diagram would be drawn as below.

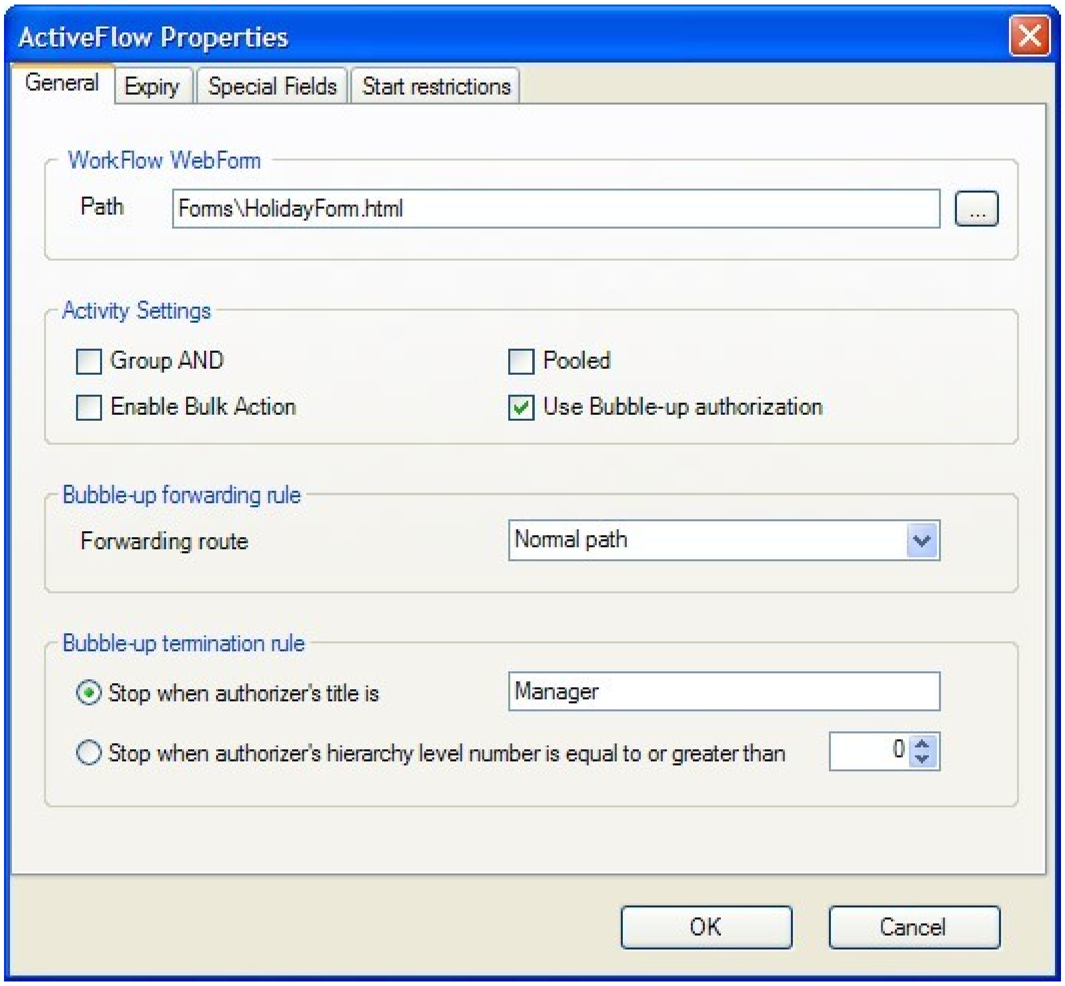


Activity	Description
Holiday Request	The form will bubble up within department up to department manager (the stop condition for the bubble up is Title="Manager").
Check Values	After department manager's approval the form has to be checked by a person from the Human Resources department.

Note:

We placed the "Holiday Request" activity in **Any** department and **Any** role because "Any" is a keyword and allows any user to start this workflow. For more information regarding restrictions of starting a new workflow please check the Start restrictions chapter.

Set the bubble-up routing type for the department of the "Holiday Request" activity. For this, right click on the task and select **Settings...** from **ActiveFlow** menu. In the **General** tab check the Use Bubble-up authorization, select the forwarding route and termination rule as below:



WorkFlow WebForm

Path: Forms\HolidayForm.html

Activity Settings

- [ ] Group AND
- [ ] Pooled
- [ ] Enable Bulk Action
- [x] Use Bubble-up authorization

Bubble-up forwarding rule

Forwarding route: Normal path

Bubble-up termination rule

[✓ Stop when authorizer's title is] Manager

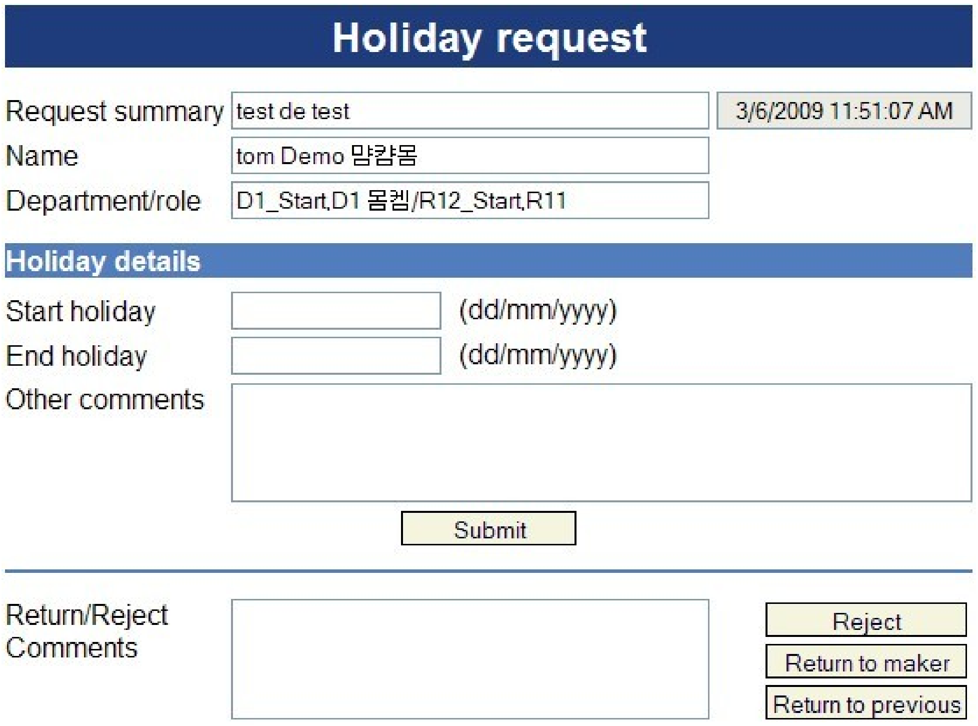
Stop when authorizer's hierarchy level number is equal to or greater than 0

Step 3 - Design the workflow form

The workflows forms can be designed using any HTML editor. The form design is beyond the purpose of this document as we assume the designer has knowledge of HTML or ASP.NET and C#. For more details regarding form design constraints please consult the Workflow forms chapter.

Note:

A nice and easy to use form is very important for any workflow as this is what the end-user will see. But for the purpose of this example the most basic form could look like this :



Holiday request

Request summary	test de test	3/6/2009 11:51:07 AM
Name	tom Demo 먛캳몸	
Department/role	D1_Start,D1 몸켐/R12_Start,R11	

Holiday details

Start holiday		(dd/mm/yyyy)
End holiday		(dd/mm/yyyy)
Other comments		

The HTML code generated automatically by the HTML editor for this form is as below:

```
<html>
<head></head>
<body>
  <form>
  <style type="text/css">
    #body
    {
      font-family: Arial;
      font-size: 12px;
    }
    .T_TableMain
    {
      width: 520px;
    }
    .T_Caption
    {
      background-color: #517dbf;
      color: #fff;
      font-weight: bolder;
    }
    .FormTitle
    {
      font-size: 24px;
      padding-right: 3px;
      padding-left: 3px;
      font-weight: bold;
      padding-bottom: 3px;
      color: #fff;
      padding-top: 3px;
      background-color: #1e3c7b;
      margin-bottom: 10px;
    }
    .inputBtn
    {
      border: solid 1px #000;
      padding: 1px;
      cursor: pointer;
      color: #000033;
      background-color: #F4F4DF;
    }
    .description
    {
      padding-left: 10px;
    }
  </style>
  <script type="text/javascript">
    function SetDefaultValues(){
      // for maker, initialize the empName and empPosition fields
      var bMaker = <%=AF_GetCurrentUserType()%>;
      if(bMaker == 0)
      {
          document.forms[0].crtDate.value = "<%=CStr(Now)%> "
          document.forms[0].empID.value = AF_UserID;
          document.forms[0].empName.value = AF_CrtFirstName + " " + AF_CrtLastName
          document.forms[0].empPosition.value = AF_CrtDepartmentName + "/" + AF_CrtRoleName;
      }
    }
  </script>
  <%
  ' update HR database function
  function UpdateHoliday(sUserID, sStartDate, sEndDate)
    ' write here your custom code
    UpdateHoliday = True
  end function
  %>
  <table align="center" class="T_TableMain">
    <tr>
      <td colspan="3">
          <div align="center" class="FormTitle">Holiday request</div>
      </td>
    </tr>
    <tr>
      <td width="20%" style="white-space: nowrap;">Request summary</td>
      <td width="65%">
          <input name="_AFFormTitle" style="width: 100%" />
```

```
      </td>
      <td width="15%" align="right" style="white-space: nowrap;">
        <input name="crtDate" readonly="readonly" style="text-align: center;" />
      </td>
    </tr>
    <tr>
      <td>Name</td>
      <td>
        <input id="empName" name="empName" style="width: 100%" />
        <input type="hidden" id="empID" name="empID" />
      </td>
    </tr>
    <tr>
      <td>Department/role</td>
      <td><input name="empPosition" style="width: 100%" /></td>
    </tr>
    <tr>
      <td colspan="3" nowrap="nowrap">
        <div class="T_Caption" style="margin-top: 10px; margin-bottom: 4px;">Holiday details</div>
      </td>
    </tr>
    <tr>
      <td>Start holiday</td>
      <td>
        <input name="startDate" style="width: 120px"><span class="description">(dd/mm/yyyy)</span>
      </td>
    </tr>
    <tr>
      <td>End holiday</td>
      <td>
        <input name="endDate" style="width: 120px"><span class="description">(dd/mm/yyyy)</span>
      </td>
    </tr>
    <tr>
      <td valign="top">Other comments</td>
      <td colspan="2"><textarea cols="50" name="requestComments" rows="3"></textarea></td>
    </tr>
    <tr>
      <td colspan="3" align="center">
        <input class="inputBtn" name="B1" type="submit" value="Submit" style="width: 100px;
          height: 20px" />
      </td>
    </tr>
    <tr>
      <td colspan="3">
        <div class="T_Caption" style="height: 2px; margin-top: 10px; margin-bottom: 10px;"></div>
      </td>
    </tr>
    <tr>
      <td valign="top">Return/Reject Comments</td>
      <td width="50%" valign="top">
        <textarea name="_AFRRComments" style="width: 100%; height: 70px" id="Textarea1"></textarea>
        <input id="_AFRejectReturn" name="_AFRejectReturn" type="hidden" value="OFF" />
        <input id="_AFRR" name="_AFRR" type="hidden" />
```

```
            <script type="text/javascript">
               function _AFSetAction(action) {
                  document.getElementById("_AFRejectReturn").value = "ON";
                  document.getElementById("_AFRR").value = action;
               }
            </script>
         </td>
         <td width="25%">
            <table width="100%" style="border-right: #cc3300 0px solid; border-top: #cc3300 0px solid;
               border-left: #cc3300 0px solid; border-bottom: #cc3300 0px solid" id="Table1">
               <tr>
                  <td align="right">
                     <input class="inputBtn" name="_AFSubmitReject" type="submit" style="height: 20px;
                        width: 115px" value="Reject" onclick='_AFSetAction("_AFReject")' id="Submit4" />
                  </td>
               </tr>
               <tr>
                  <td align="right">
                     <input class="inputBtn" name="_AFSubmitReturnM" type="submit" style="height: 20px;
                  width: 115px" value="Return to maker" onclick='_AFSetAction("_AFReturnM")' id="Submit5" />
                  </td>
               </tr>
               <tr>
                  <td align="right">
                     <input class="inputBtn" name="_AFSubmitReturnP" type="submit" style="height: 20px;
               width: 115px" value="Return to previous" onclick='_AFSetAction("_AFReturnP")'  id="Submit6" />
                  </td>
               </tr>
            </table>
         </td>
      </tr>
   </table>
   </form>
</body>
</html>
```

Save this HTML code in the *HolidayForm.html* file.

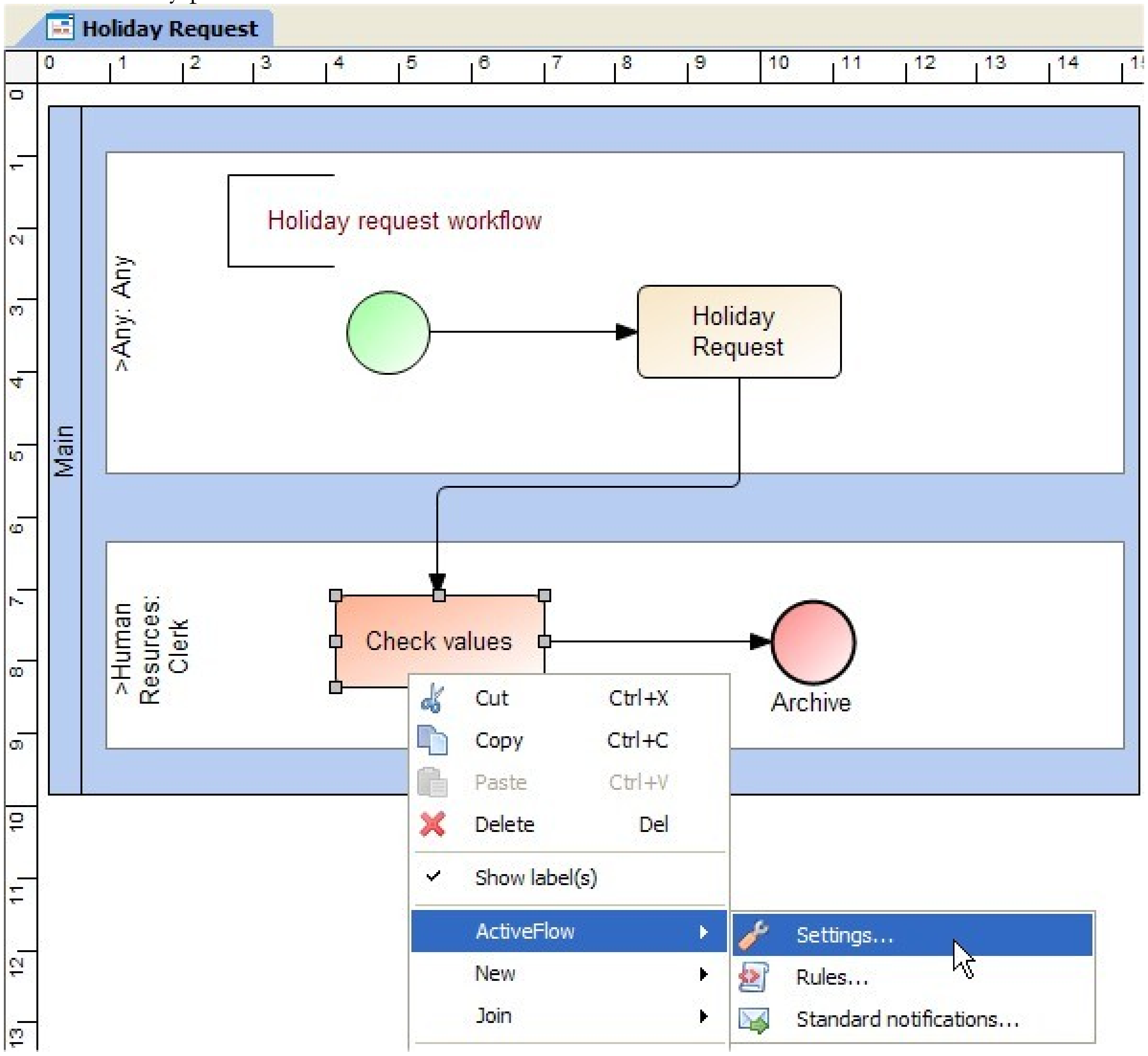
Now we will discuss the relevant parts of the form. We have shown them with colored backgrounds for the purpose of this discussion..

1. The form tag - yellow background - all forms must have a form tag.
2. The STYLE section - blue background - defines the styles used for controls in this form. A better approach would be to create a Cascading Style Sheet file (css) and include it in the workflow forms. This way will ensure that all the forms will have the same "look and feel" and you will be able to modify their appearance by simply editing the css file.
3. The custom client side JavaScript - green background - is used for setting the default values for the maker. ActiveFlow calls the function SetDefaultValues for setting the default form values. The function is called **after** ActiveFlow fills the form with values sent by previous users. For more details about customization and the ActiveFlow API please check the documentation.

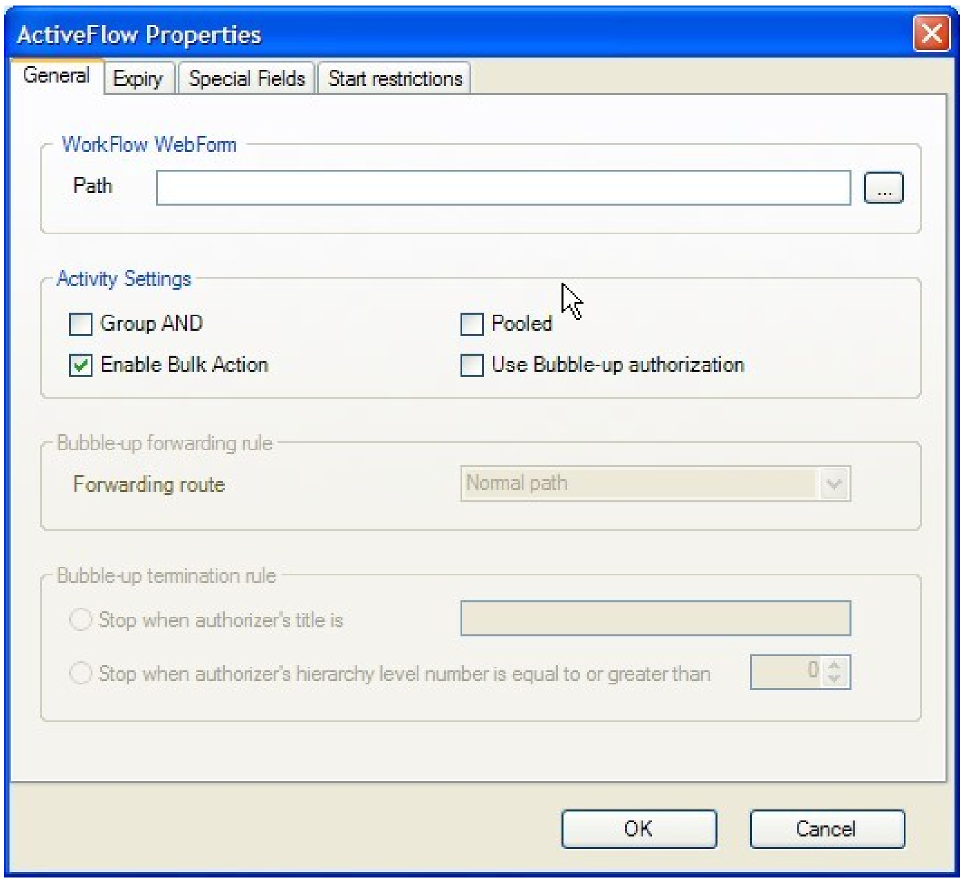
4. Server side custom script – light yellow color - the function for updating the HR database.
5. Title AF Control and RejectReturn AF Control - orange background - This is the code generated by the AF controls.
6. The submit button - gray background - all forms **must** have a submit button.

Step 4 - Attach forms to the activities

After designing the workflow form(s) you will need to attach them to activities in the diagram. You can attach the same form for all activities or different form for each activity. In our case, we'll attach the same form to each activity. Select all activities by clicking on them and keeping the SHIFT key pressed.



Right click on any activity and select **Settings...** from the **ActiveFlow** menu. In the dialog that appears, press the button on the right side to browse for the form and choose the *HolidayForm.html* .

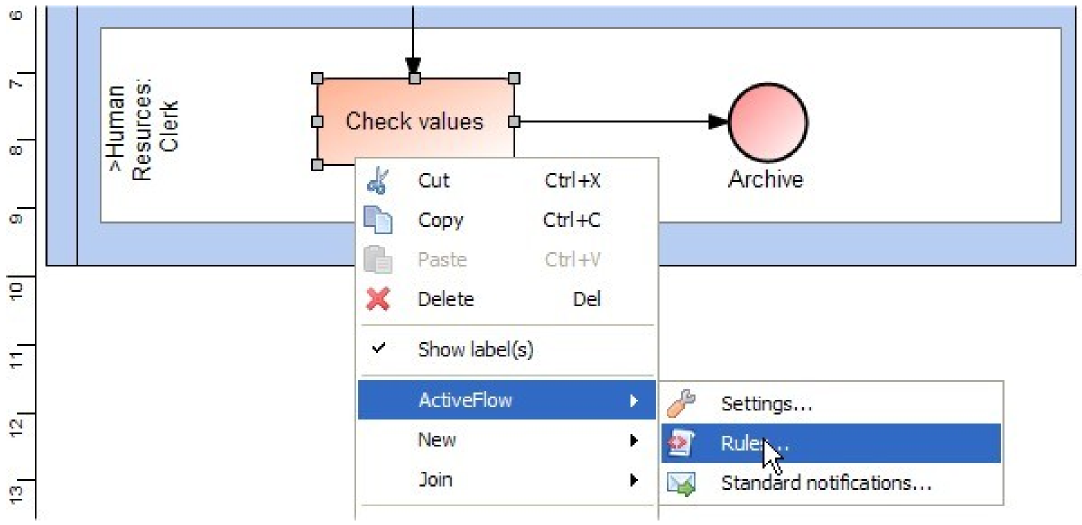

ActiveFlow Properties
General
Expiry
Special Fields
Start restrictions
WorkFlow WebForm
Path
...
Activity Settings
Bubble-up forwarding rule
Forwarding route
Normal path
Bubble-up termination rule
Stop when authorizer's title is
Stop when authorizer's hierarchy level number is equal to or greater than
0
OK
Cancel

Group AND [ ]
Pooled [ ]
Enable Bulk Action [✓]
Use Bubble-up authorization [ ]

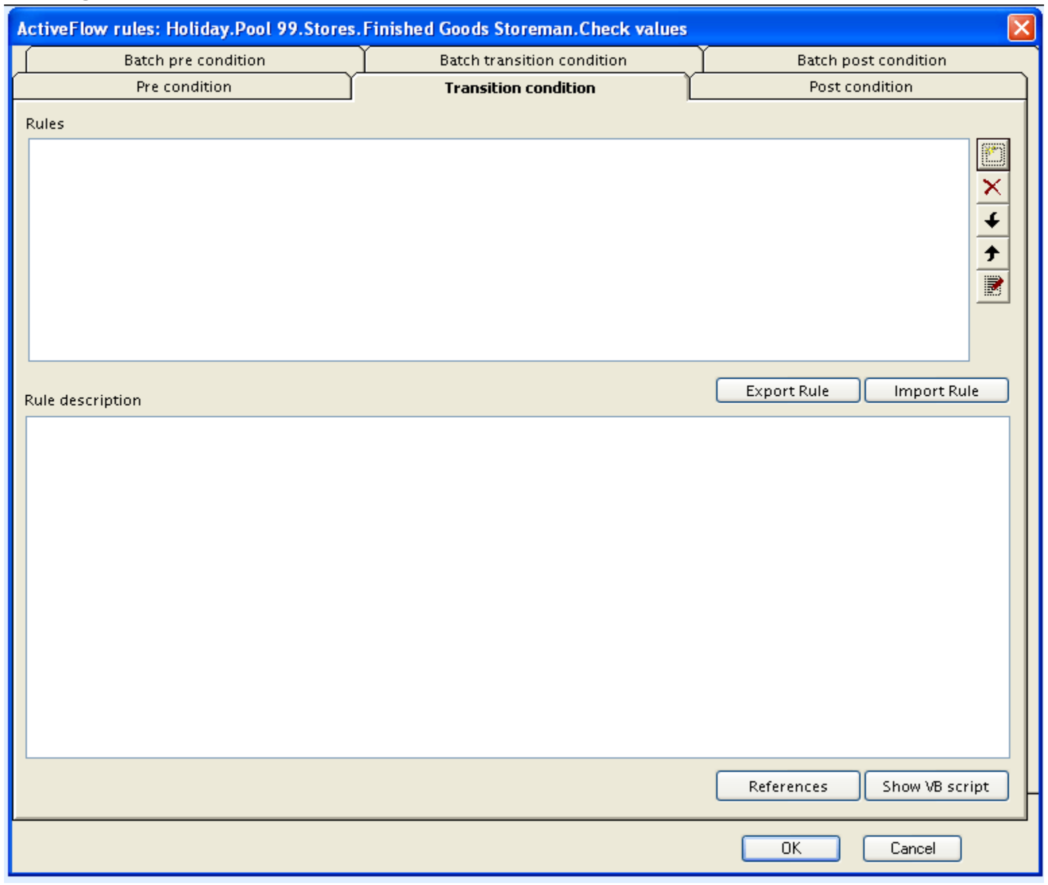
Step 5 - Define transition actions (optional)

If you want to update external systems, now is the time to write the custom script. For this particular workflow you might want to update a Human Resources database with the holiday period for this user. We assume that there is a function which does this. The function would take as parameters the userID, start date and end date and you have to define it either in an external file which will be included in the workflow form, or in the form itself. We recommend the first option as you won't need to re-run the Workflow Wizard every time you modify the function.

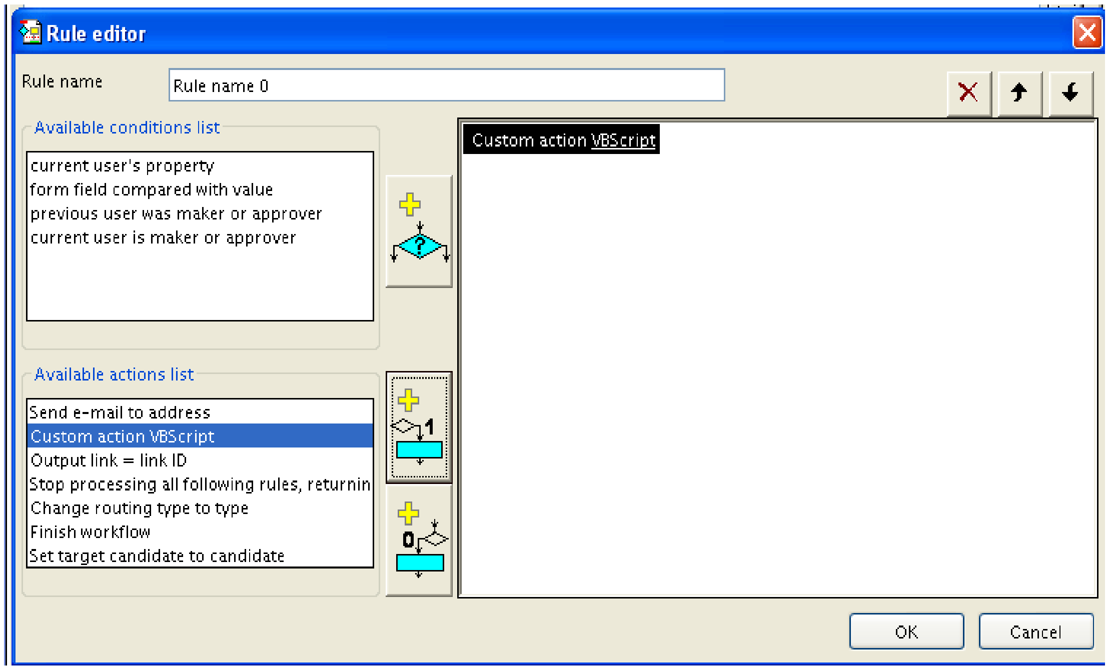
You will define this action for the **Check values** activity. Right click on the activity and select **Rules...** from the **ActiveFlow** menu .



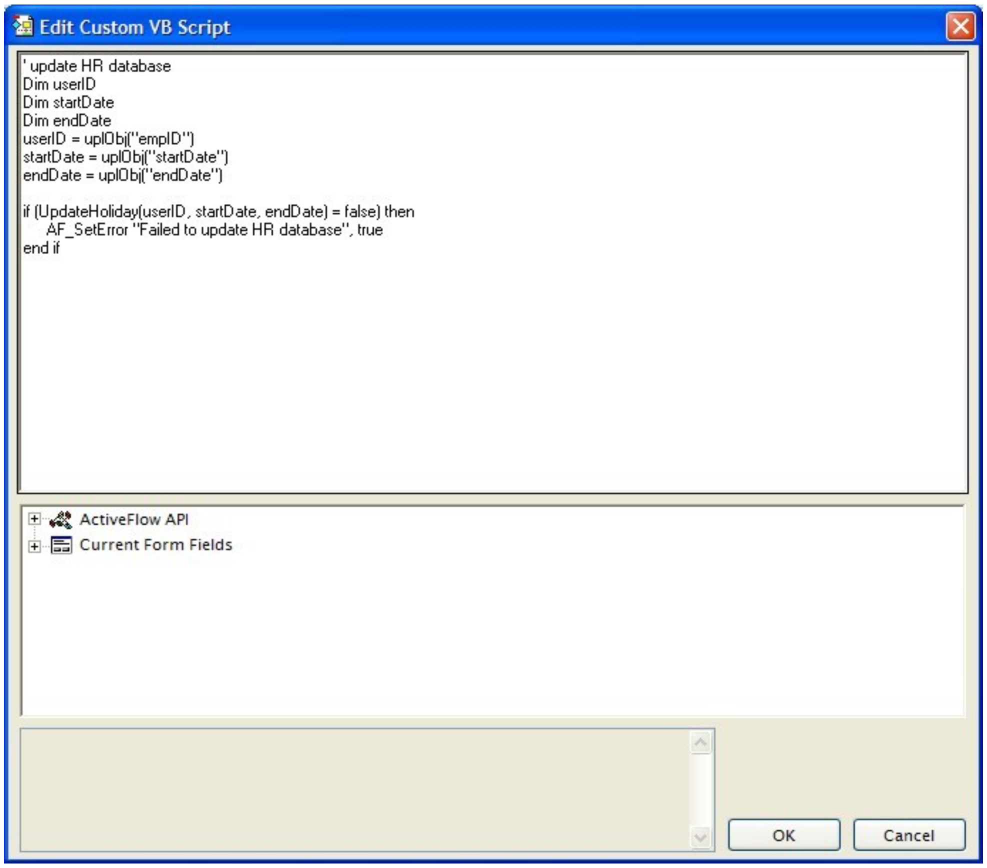
In the rules wizard dialog, select the **Transition condition** tab and press the **Add new rule** button on right side.



In the **Available actions list** select "*Custom action VBScript*" and press the **Add action** button. Then, on the right panel, press the underlined VBScript text.


Rule editor
Rule name
Rule name 0
Available conditions list
current user's property
form field compared with value
previous user was maker or approver
current user is maker or approver
Available actions list
Send e-mail to address
Custom action VBScript
Output link = link ID
Stop processing all following rules, returnin
Change routing type to type
Finish workflow
Set target candidate to candidate
Custom action VBScript
OK
Cancel

In the newly opened window you will need to write the code for updating Human Resources database as below.



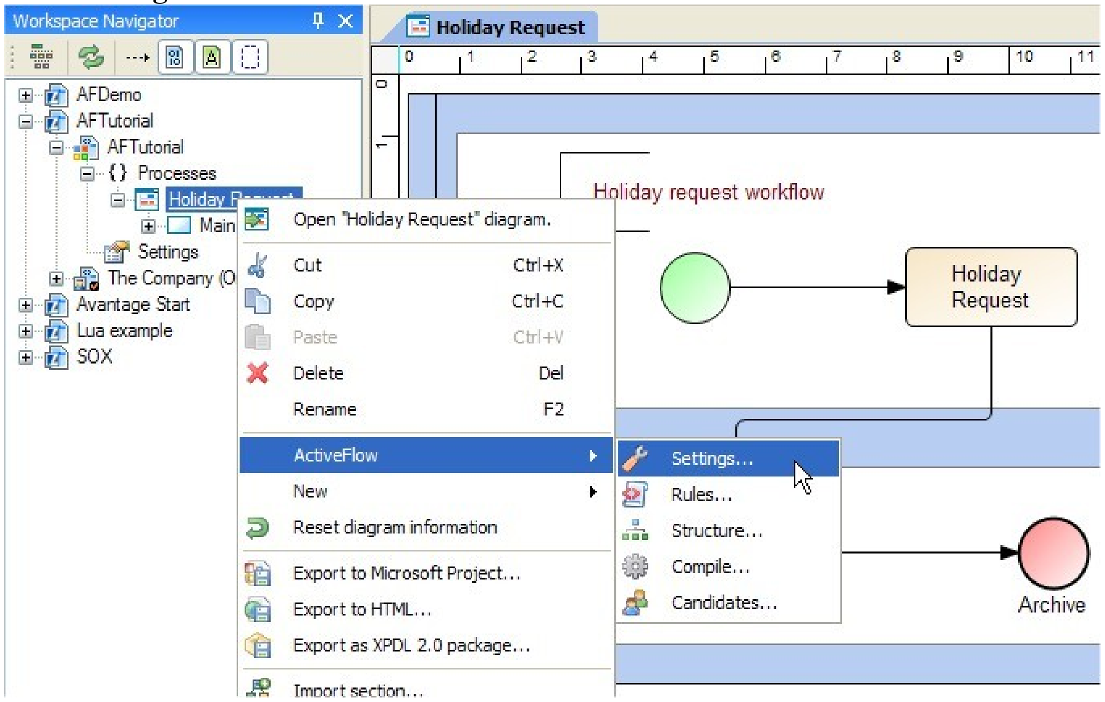
```
' update HR database
Dim userID
Dim startDate
Dim endDate
userID = uplObj("empID")
startDate = uplObj("startDate")
endDate = uplObj("endDate")

if (UpdateHoliday(userID, startDate, endDate) = false) then
    AF_SetError "Failed to update HR database", true
end if
```

Note:

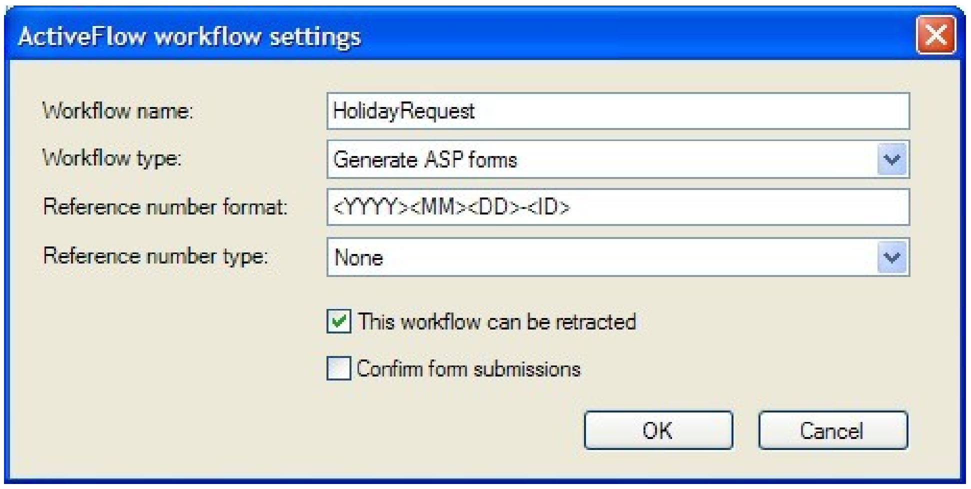
The function **UpdateHoliday** has to be defined in a file included as a reference in the Workflow rules. For more details please check the ActiveFlow Rules dialog section.

Step 6 - Set the workflow properties

The next step would be to set the workflow name and the location where the Workflow Wizard will generate the files. To do so, right-click on "Holiday Request" diagram from BPTree and select the **Settings...** from **ActiveFlow** menu.



Then in the **ActiveFlow settings** dialog enter the values as below:



ActiveFlow workflow settings

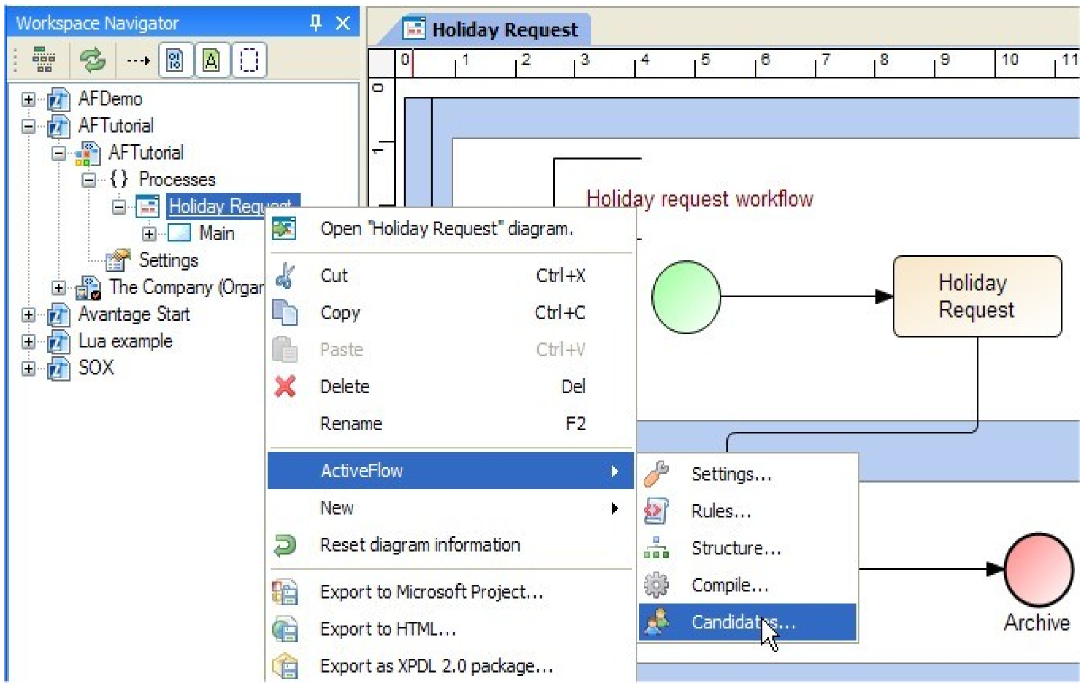
Workflow name:	HolidayRequest
Workflow type:	Generate ASP forms
Reference number format:	<YYYY><MM><DD>-<ID>
Reference number type:	None

[✓] This workflow can be retracted

[ ] Confirm form submissions

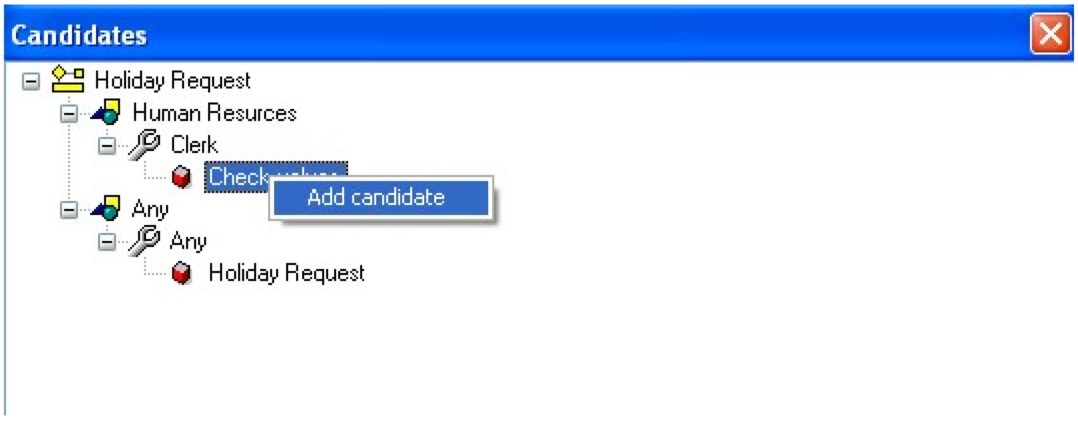
Step 7 - Set the workflow candidates

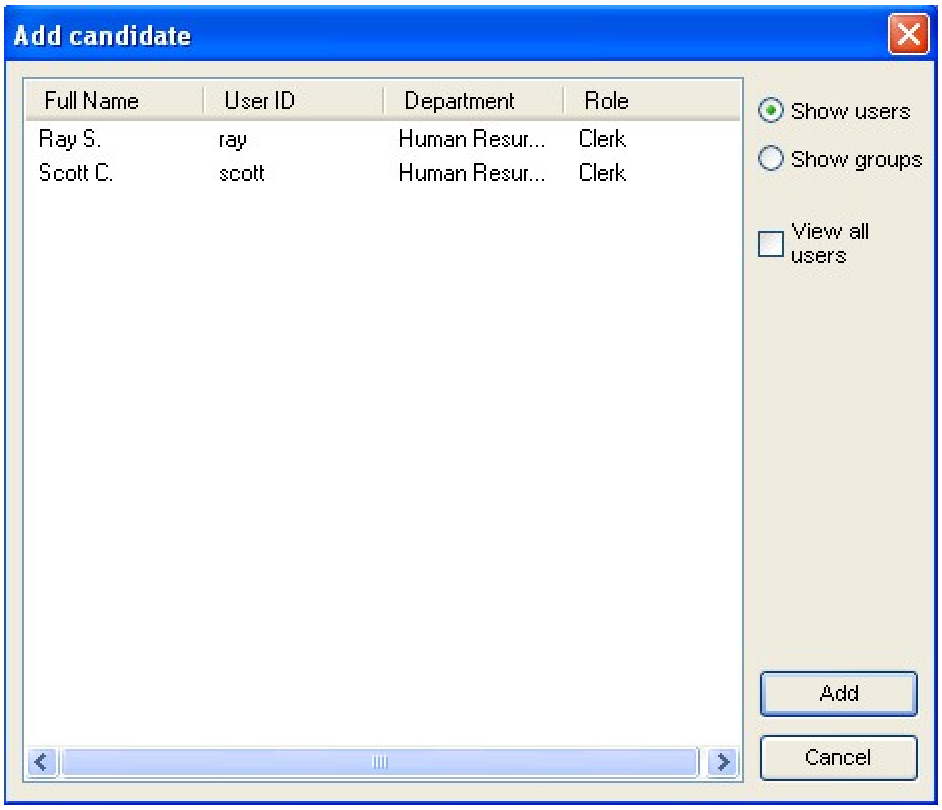
Right-click on “Holiday request” diagram from BPTree and select the **Candidates....**from **ActiveFlow** menu.



The **Candidates** dialog will display the organization structure and activities in the diagram. You don't have to set candidates for the starting activity (Holiday Request) as the form can be filled in by anybody.

Now for the next activity. Right click on the **Check values** activity and select the **Add candidate** menu.



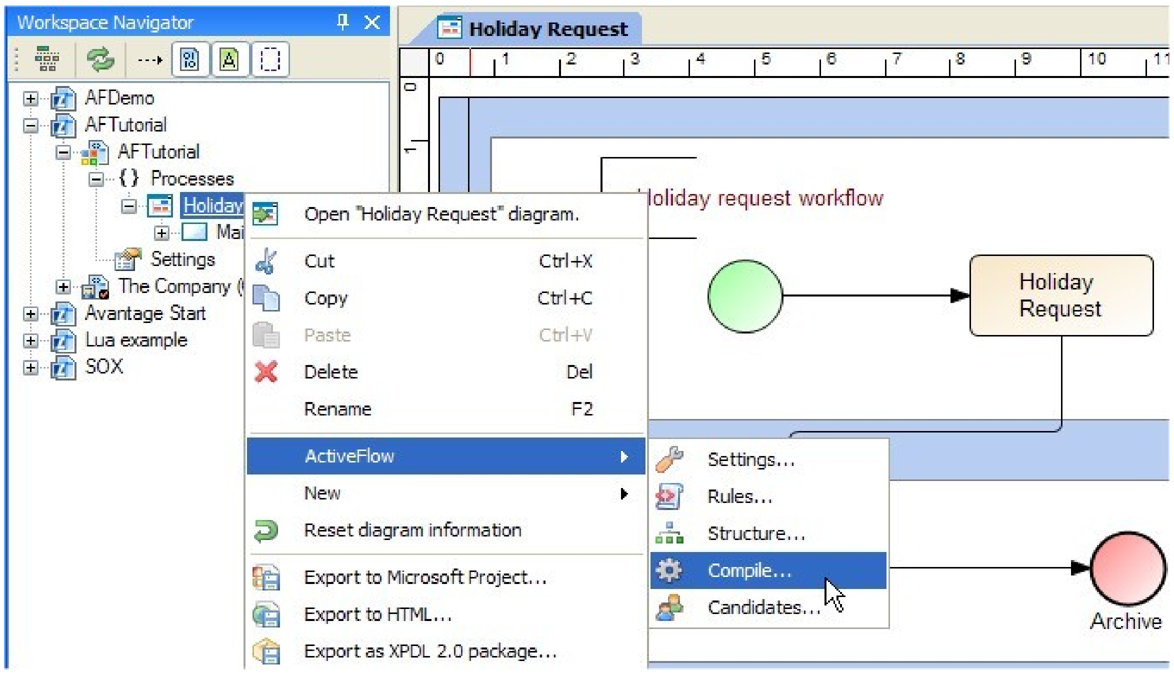


Full Name	User ID	Department	Role
Ray S.	ray	Human Resur...	Clerk
Scott C.	scott	Human Resur...	Clerk

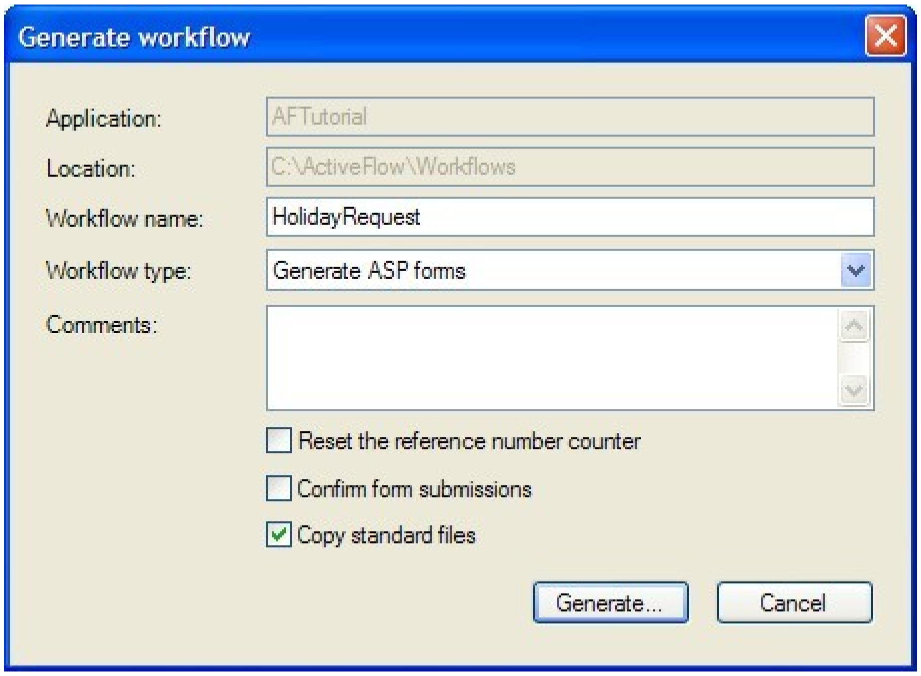
The dialog displays users under Human Resources/Clerk role (because the activity was placed under this role). Select one user and then press the **Add** button. Repeat the actions and add other users as candidates for this activity if required.
In our example, we specify that any of the 2 users can receive jobs for this activity and the ActiveFlow system will choose at run-time the user who has the least amount of work.

Step 8 - Run the workflow wizard

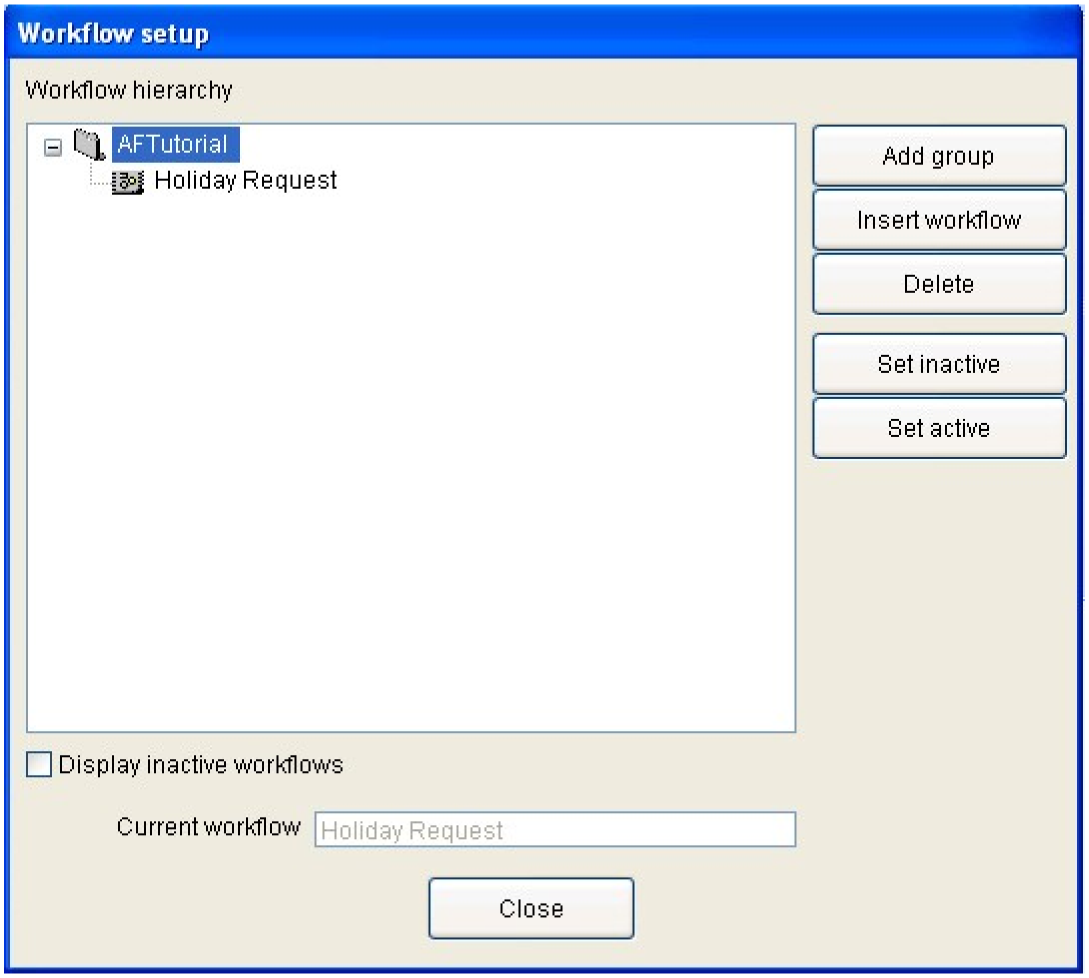
This is the final step before testing the workflow. Right-click on the Holiday Request diagram from BPTree and select the **Compile...** from **ActiveFlow** menu.



In the wizard dialog, select the **Build all** checkbox.

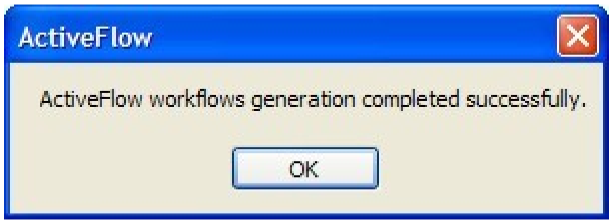


Application: AFTutorial
Location: C:\ActiveFlow\Workflows
Workflow name: HolidayRequest
Workflow type: Generate ASP forms
Comments:
☐ Reset the reference number counter
☐ Confirm form submissions
☑ Copy standard files



Current workflow: Holiday Request

Press **Insert workflow** button. Then press the **Close** button and wait for the wizard to complete.



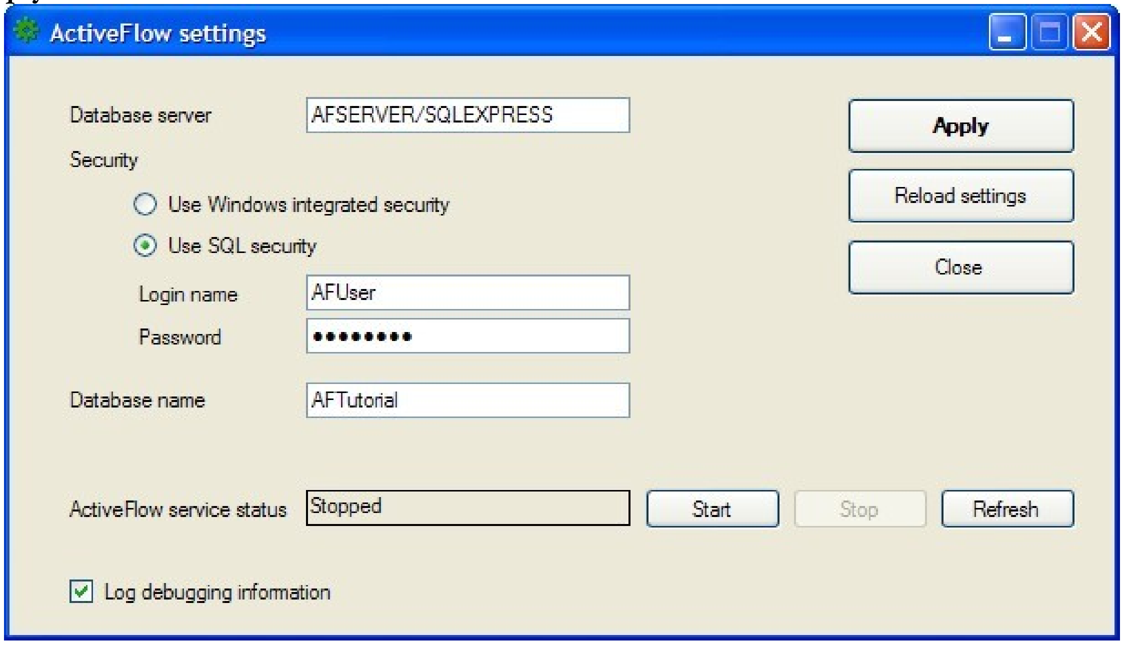
ActiveFlow workflows generation completed successfully.

Press OK button.

Step 9 - Test the workflow

The last step would be to test the workflow. We need to set up the ActiveFlow engine for this.

Select Start ->Programs-> ActiveFlow->ActiveFlow Server Settings . Fill the fields and click **Apply**:


ActiveFlow settings

Database server: AFSERVER/SQLEXPRESS

Security

○ Use Windows integrated security

● Use SQL security

Login name: AFUser

Password: ••••••••

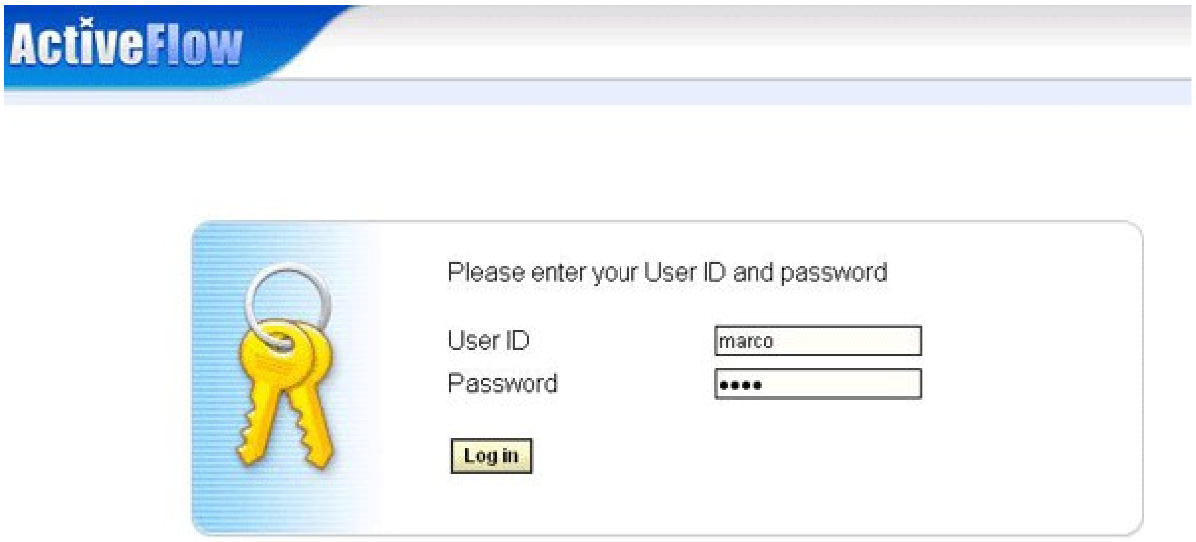
Database name: AFTutorial

ActiveFlow service status: Stopped

☑ Log debugging information

TEST THE WORKFLOW:

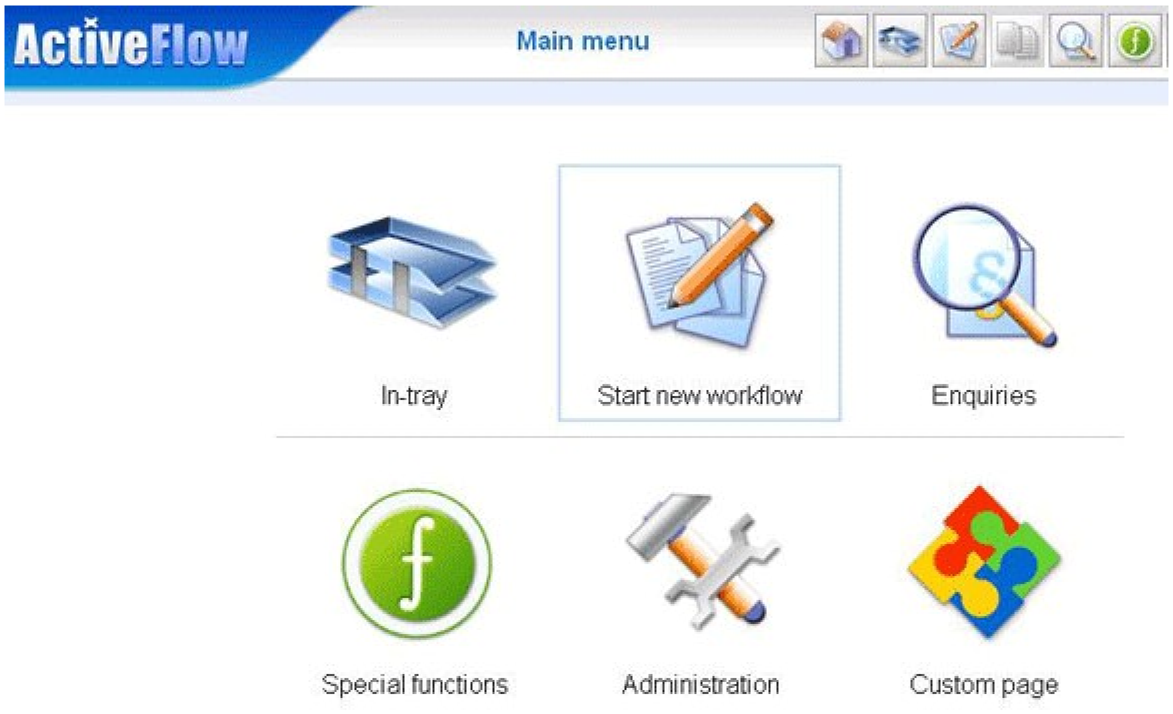
- Start Internet Explorer (IE) and in the address bar type: http://AFSERVER/AFTutorial/


Please enter your User ID and password

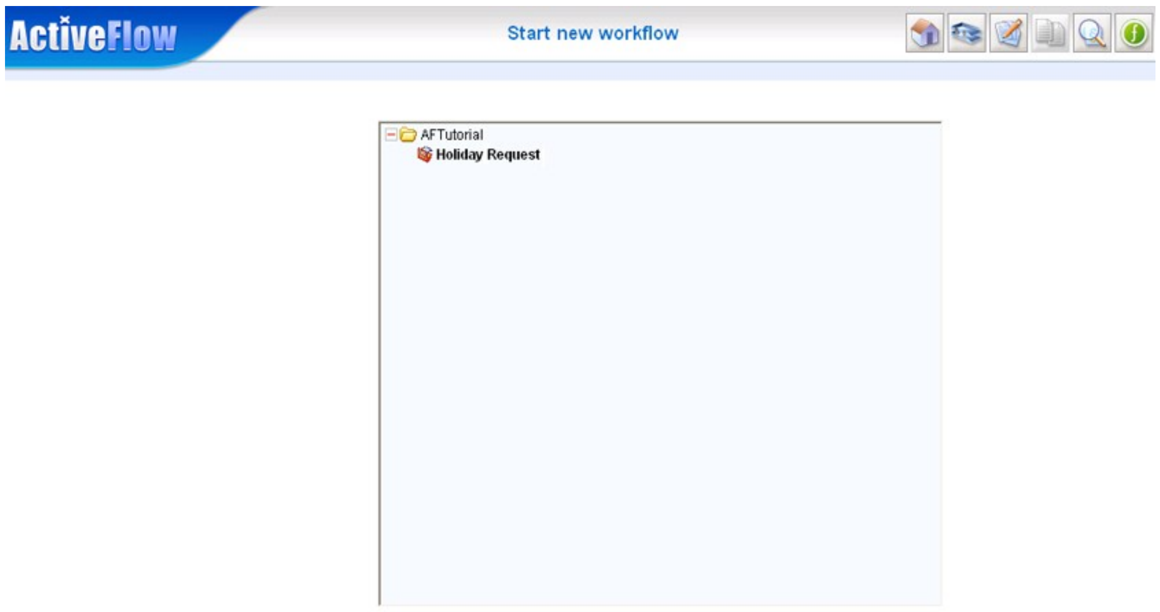
User ID: marco

Password: ••••

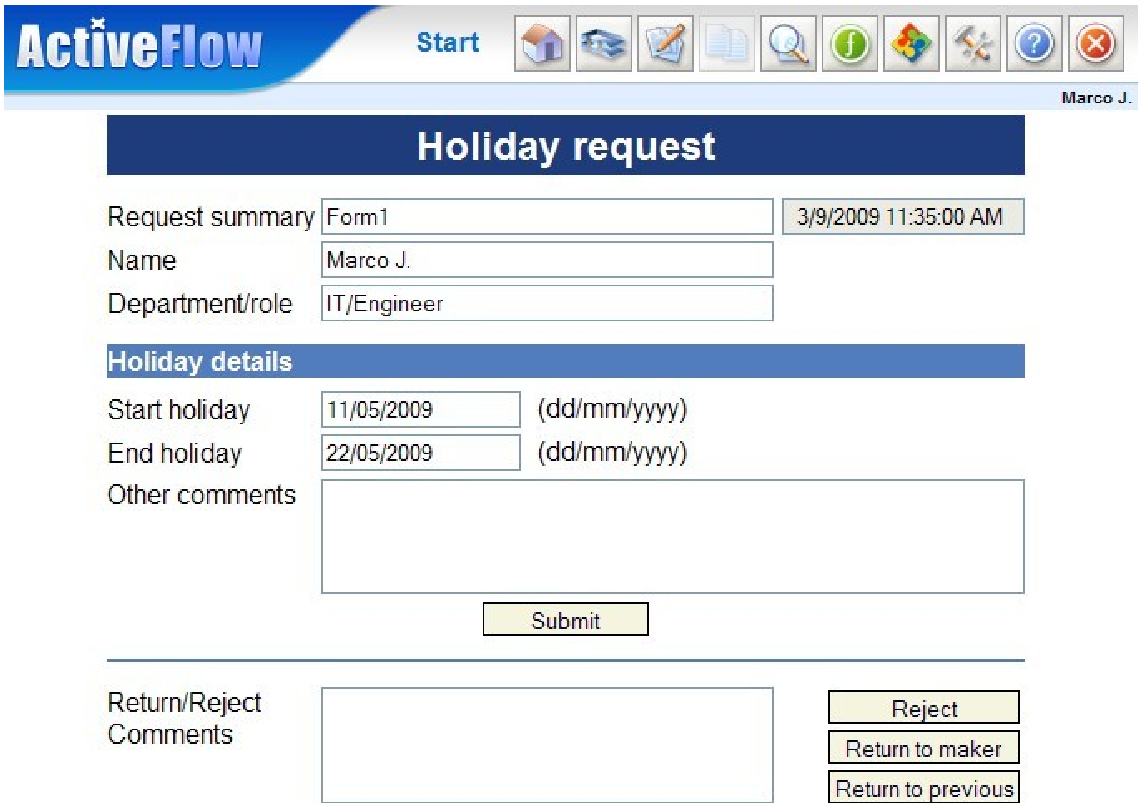
- Log on as J. Marco (userID: marco, password: demo)



- Press the **Start new workflow** button



- Select the **Holiday Request** workflow



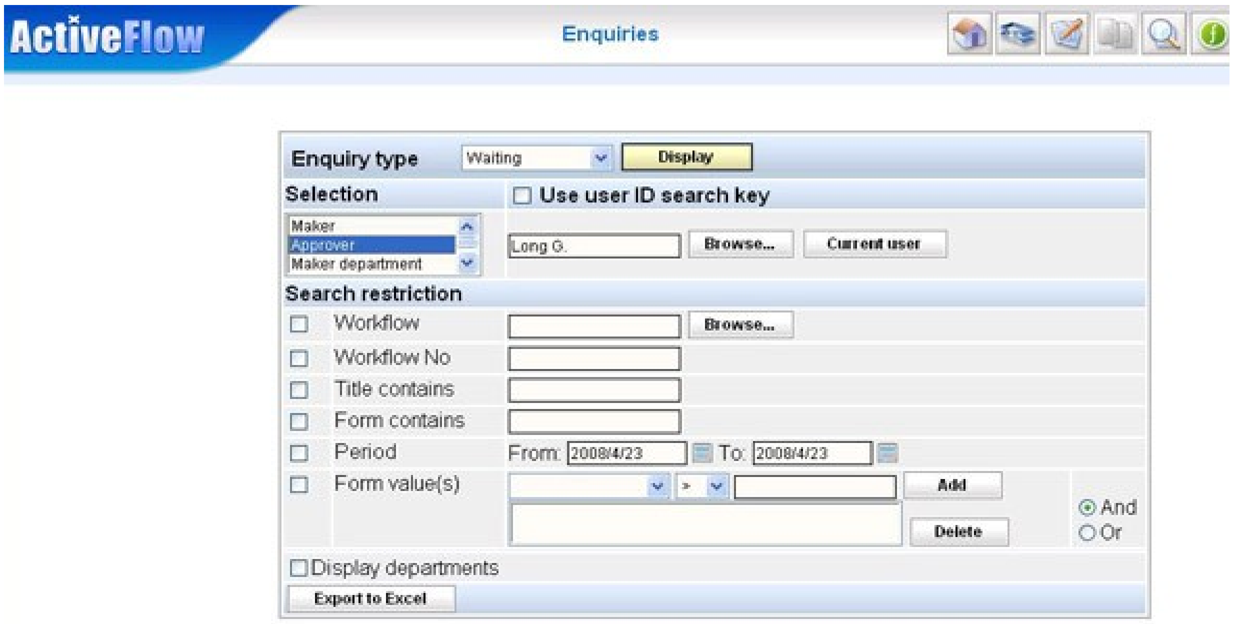
Holiday request

Request summary	Form1	3/9/2009 11:35:00 AM
Name	Marco J.	
Department/role	IT/Engineer	

Holiday details

Start holiday	11/05/2009	(dd/mm/yyyy)
End holiday	22/05/2009	(dd/mm/yyyy)
Other comments		

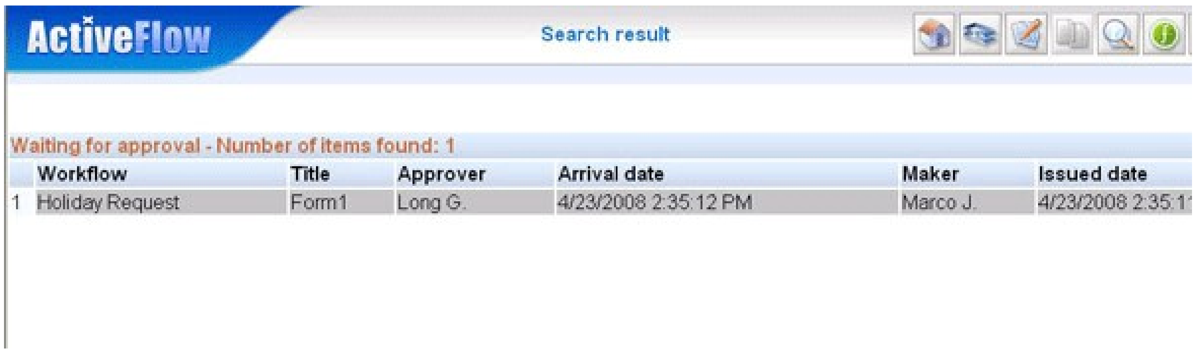
- Fill the form fields and press **Submit**.
- If no error is reported it means the form was submitted to J. Marco's supervisor (normal route user): G.Long.
- In the browser window, press the **Logout** button (the right most button from the ActiveFlow headbar - the one with the Exit door) and next log on as G.Long (userID: long, password: demo).
- From the main menu page press the **Enquiries** button. The Enquiries page is displayed as below.



Enquiries

Enquiry type	Waiting	Display	
Selection	☐ Use user ID search key		
Maker / Approver / Maker department	Long G.	Browse...	Current user
Search restriction			
☐ Workflow		Browse...	
☐ Workflow No			
☐ Title contains			
☐ Form contains			
☐ Period	From: 2008/4/23	To: 2008/4/23	
☐ Form value(s)	>	Add / Delete	[✓ And] Or
☐ Display departments			
Export to Excel			

- Select the **Display** button.



Search result

Waiting for approval - Number of items found: 1

	Workflow	Title	Approver	Arrival date	Maker	Issued date
1	Holiday Request	Form1	Long G.	4/23/2008 2:35:12 PM	Marco J.	4/23/2008 2:35:1

- Click on the Holiday Request workflow. The form will be displayed filled with data sent by the previous user.
- Press the **Submit** button in order to send the form forward. As G.Long's title is "Manager", the bubble-up routing will stop at this stage and the form will be send to the one of candidates of the **Check values** activity (S.Ray or C.Scott). The ActiveFlow system will choose at run-time the appropriate user. In our case the form was sent to Scott.
- Logout from the G.Long account and log in as Scott (userID: scott, password: demo).
- Open the **In-tray** page and view the form sent by G.Long.
- Press **Submit** in order to finally approve the request.

Well that's it and the form has now been finally approved. The form is kept on system in a live archive form and we can view the Reports if we want to trace the workflow at a later stage. If required the Human Resources database will have been updated as well.

This sample can be found in the ***Avantage Projects \AFTutorial .***

ActiveFlow API

The APIs described here can be used in your html/asp workflow forms in order to get important information about a workflow, e.g. The type of the form opened by the user, the status of the workflow, etc. Sometimes the workflow designer must know the type of user who opens the form. For example, some fields should filled only by the maker and for all the users they should be read-only.

Name	Description
AF_GetDisplayMode	Returns whether ActiveFlow will display the form in read-only mode (enquiries or Cc forms) or not.
AF_GetCurrentUserType	Retrieves the type (maker or approver) of the user who opened the form.
AF_GetSourceUserType	Retrieves the type (maker or approver) of the user who sent the form to the current user.
AF_GetSentFieldValue	Useful for retrieving the values sent by a user. The designer should use this function in case it is necessary to customize the cancel workflow actions.
AF_TraceFlowBack	Returns the list of previous approvers.
AF_GetFlowStatus	Returns the status of the forms in the current flow (e.g. who approved/returned/issued the form, where the form is waiting for approval, etc).
AF_GetAction	Returns the type of the action performed by the user (approved,returned,rejected,held).
AF_GetUserCustomAttribute	Returns the user's custom attribute.
AF_SetUserCustomAttribute	Sets the user's custom attribute.
AF_GetDepartment	Returns the current user's department name or CODE.
AF_GetDepartmentID	Returns the department't ID for a given CODE.
AF_GetBubbleUp	Returns the normal or alternative route of the specified user.
AF_GetUsersInDepartment	Returns a dictionary object containing the list of users in the specified department.
AF_GetUsersInRole	Returns a dictionary object containing the list of users in the specified role.
AF_GetDepartmentName	Returns the name of the department specified by CODE.

AF_GetRoleName	Returns the name of the role specified by CODE.
AF_GetParentDept	Returns the parent department of the specified department/role.
AF_GetSubDepartments	Returns a dictionary object containing the list of subdepartments of the given department.
AF_GetCurrentWorkflow	Returns the current workflow name.
AF_GetCurrentActivity	Returns the current activity name.
AF_GetCurrentActivityID	Returns the current activity ID.
AF_GetRoutingType	Returns the algorithm used by ActiveFlow in order to submit the form.
AF_GetFormType	Returns the type of the form (approval, returned, Cc, held, delegate).
AF_GetFieldValue	Returns the field value for a given job. The main purpose of this function is to access the field values from the bulk transition functions.
AF_ChangeFieldValue	Changes the contents of the specified field for the current form. Returns True if successful.
AF_SetExpireFlag	Enable/disable the expire functionality for current workflow.
AF_SetExpireFlagEx	Enable/disable the expire functionality for the specified workflow.
AF_GetMakerType	Returns the user type for starting a new workflow.
AF_GetActivityName	Returns the name of the given activity ID. The name of the Activity translated to specific language as defined in the user preference.
AF_GetBubbleUpSteps	Returns the number of bubble-up steps to get to the current user. Can be used both when the form is loaded and when the form is submited. **Note:** for final approval, use it in OnPreCondition or OnTransitionCondition.
AF_SetBubbleUpTarget	Sets a new user for the bubble-up routing
AF_SetTargetUser	Dynamically sets the target candidate for a given target. **Note:** This function should be called in OnPreCondition or OnTransitionCondition functions.

AF_GetTargetUser	Returns the userID (or list of userIDs in the case of group routing) to whom the current form has been sent. **Note:** This function should be called in OnPostCondition or OnBatchPostCondition functions.
AF_UserIsWorkflowMaker	Returns True if the given user is the maker of the current workflow.
AF_IsUserInGroup	Returns True if the given user belongs to the given group.

AF_GetDisplayMode	can be used within the <form> tags in the form.
input	
output	0 if AF tries to disable the controls 1 otherwise

AF_GetCurrentUserType	can be used within the <form> tags in the form.
input	
output	-1 = error 0 = current user is the maker 1 = current user is an approver

AF_GetSourceUserType	can be used within the <form> tags in the form.
input	
output	-1 = error or the current user is the maker 0 = source user is the maker 1 = source user is an approver

AF_GetSentFieldValue	can be used in OnPreRetract event. (see *Workflow customization/Cancel workflow functions* for more details).
input	msgGroupID = the ID of the group of messages. This is also a parameter of the OnPreRetract function so it can be used as it is. fieldName = the form field name.
output	The contents of the field as a string.

AF_TraceFlowBack	can be used within the <form> tags.
input	
output	returns a collection of collections, each row containing the following information: First Name and Last Name Date Comments Action type = -1 issued 0 approved 1 returned Handled by = 0 - user 1 - ActiveFlow expiry check system 2 - ActiveFlow robot

Example:

The following example shows how this API can be used within the <form> tags in the form attached to an activity.

```
<%
htmlStr = "<TABLE><TR>" + _
"<TD>Name</TD>" + _
"<TD>Date</TD>" + _
"<TD>Comments</TD>" + _
"<TD>Action type </TD></TR>"

Set histList = AF_TraceFlowBack
for crt = 0 to histList.Count - 1
set crtItem = histList.Item(crt)
act = ""
select case CInt(crtItem.Item(3))
case -3 act = "Finaly approved"
case -2 act = "Rejected"
case -1 act = "Issued"
case 0 act = "Approved"
case 3 act = "Approved/Re-issued"
case 1,4,5 act = "Returned"
end select
htmStr = htmStr + "<TR>" + _
"<TD>" + CStr(crtItem.Item(0)) + "</TD>" + _
"<TD>" + CStr(crtItem.Item(1)) + "</TD>" + _
"<TD>" + CStr(crtItem.Item(2)) + "</TD>" + _
 "<TD>" + act + "</TD></TR>"
next
```

```
Response.Write(htmStr + "</TABLE>")
%>
```

The items in blue may be customized by the form designer. Also the <TABLE><TR><TD> tags may be customized to have the required appearance (background color, borders, size etc).

If a workflow audit trail is to be a standard feature across several workflows, the following is recommended:

- Include a file called history.asp between the <form> tags of the file attached to the map activities
- The include is located in the place where the list of approvers is to be displayed

```
<!--#INCLUDE FILE="History.asp"--!>
```

ActiveFlow provides the history.asp file as a standard file in:

C:\ProgramFiles\Avantage\Plugins\ActiveFlowDesigner\ActiveFlow\StandardForms\StandardASP

AF_GetFlowStatus	can be used within the <form> tags.
input	

<u>output</u>	returns a collection of collections, each row containing the following information: First Name and Last Name Date Comments Status = -1 - Issued -2 - Rejected -3 - Finally approved 0 – Hold (for in-flight workflows) / Approved (for finaly archived workflows) 1 – Pending (for in-flight workflows) / Approved or Re-issued (for finaly archived workflows) 2 – Waiting (for in-flight workflows) / Returned (for finaly archived workflows) 3 – Approved (for in-flight workflows) / Returned (for finaly archived workflows) 4 - Returned Handled by = 0 - user 1 - ActiveFlow expiry check system 2 - ActiveFlow robot Type =-1 - Issued 0 - Normal 1 - Returned (still waiting - status is 0 or 2) 2 - Normal job returned 3 - Returned job returned again 5 - Cc

<u>Example:</u>

The following example shows how this API can be used within the <form> tags in the form attached to an activity.

```
<%
htmlStr = "<TABLE><TR>" + _
"<TD>Name</TD>" + _
"<TD>Date</TD>" + _
"<TD>Form status</TD></TR>"

Set histList = AF_GetFlowStatus
if(Request.QueryString("Action") = "DISPLAY_RO_DATA") and
(Request.QueryString("JobType") = 1) then
bArchive = True
else
 bArchive = False
end if
```

```
for crt = 0 to histList.Count - 1
  set crtItem = histList.Item(crt)
  act = ""
   if(bArchive) then
      select case CInt(crtItem.Item(2))
        case -3 : act="Finaly approved"
        case -2 : act="Rejected"
        case -1 : act="Issued"
        case 0 : act="Approved"
        case 1 : act="Approved/Re-issued"
        case 2,3 : act="Returned"
    end select
    else
     select case CInt(crtItem.Item(3))
       case -1 act = "Issued "
       case 0 act = "Approved "
       case 1 act = "Returned "
      end select
   end if
   htmStr = htmStr + "<TR>" + _
     "<TD>" + CStr(crtItem.Item(0)) + "</TD>" + _
     "<TD>" + CStr(crtItem.Item(1)) + "</TD>" + _
     "<TD>" + CStr(crtItem.Item(2)) + "</TD>" + _
     "<TD>" + act + "</TD></TR>"
next
Response.Write(htmStr + "</TABLE>")
%>
```

The items in blue may be customized by the form designer. Also the <TABLE><TR><TD> tags may be customized to have the required appearance (background color, borders, size etc).

If the workflow status is to be a standard feature across several workflows, the following is recommended:

- Include a file called FlowStatus.asp between the <form> tags of the file attached to the map activities
- The include is located in the place where the list of approvers is to be displayed

<!--#INCLUDE FILE= "FlowStatus.asp"--!>

ActiveFlow provides the FlowStatus.asp file as a standard file in:

C:\ProgramFiles\Avantage\Plugins\ActiveFlowDesigner\ActiveFlow\StandardForms\StandardASP directory.

AF_GetAction	can be used in the *transition* functions (OnPreCondition, OnTransitionCondition, OnPostCondition) of an activity.

input	the upload object (the variable name from the transition functions is *uplObj*)
output	-1 = unknown 0 = the form was submitted 1 = the form was returned to maker 2 = the form was returned to previous 3 = the form was rejected 4 = the form was held

AF_GetUserCustomAttribute	

input	sUser = ActiveFlow username
	nIdx = attribute index (1 based)

output	a string containing the user's attribute

AF_SetUserCustomAttribute	

input	sUser = ActiveFlow username
	nIdx = attribute index (1 based)
	sValue = the new value of the attribute

output	True = if the value was correctly set False = in the case of error

AF_GetDepartment	
input	nFlag = 0 returns the department name otherwise returns the department CODE
output	department name or CODE

AF_GetDepartmentIDID	
input	sCode = the CODE of the requested department.
output	department ID

AF_GetBubbleUp	

input	sUser = the ActiveFlow username nFlag = 0 returns the normal bubble-up route of the specified user. = 1 returns the alternative bubble-up route of the specified user.
output	the normal or alternative bubble-up route

Note:

The currrent user may be accesses using the global variable "AF_UserID". The following example returns the normal route of the current user:

sNormalRoute = AF_GetBubbbleUp(AF_UserID, 0)

AF_GetUsersInDepartment	
input	sCODE = the department CODE
output	a dictionary object containing the list of usernames in the specified department.

Note:

Each item in the dictionary object has as key the index (0 based). The following example displays all the users in the specified department.

```
Set usrList = AF_GetUsersInDepartment("COD-DEPT")
for each sUser in usrList .Items
   Response.Write("<BR>User = " + CStr(sUser))
next
```

AF_GetUsersInRole	
input	sCODE = the role CODE
output	the dictionary object containing the list of usernames in the specified role.

AF_GetDepartmentName	
input	sCODE = the department CODE
output	the department name.

AF_GetRoleName	

input	sCODE = the role CODE
output	The role name.

AF_GetParentDept	

input	sCODE = the department/role CODE
	nFlag = 0 - the sCODE is a department CODE
	otherwise - the sCODe is a role CODE

output	the department CODE.

AF_GetSubDepartments	
input	sCODE = the department CODE
output	a dictionary object containing the list of the sub-departments CODE.

Note: each item in the dictionary object has as key the index (0 based).

AF_GetCurrentWorkflow	
input	
output	the current workflow name.

AF_GetCurrentActivity	
input	
output	the current activity name.

AF_GetCurrentActivityID	
input	
output	the current activity ID.

AF_GetRoutingType	
input	
output	0 = bubble-up using normal route 1 = bubble-up using alternative route 3 = match candidate

AF_GetFormType	
input	
output	0 = normal (approval) 1 = returned 2 = held 3 = delegate 4 = Cc

AF_GetFieldValue	should be used **only** from bulk transition functions.
input	sJobID = the id of the job. This parameter is passed to all bulk transition functions as a variable with name *jobID*. sFieldName = the name of the field which is to be retrieved.
output	the field value

AF_ChangeFieldValue	can be used **only** in OnTransitionCondition function.
input	sFieldName = the name of the field which is to be changed. sFieldValue = the new field value.
output	True if successful.

AF_SetExpireFlag	
input	bflag = True - the jobs in current workflow can expire and False otherwise.
output	

AF_SetExpireFlagEx	
input	bFlag = True - the jobs in specified workflow can expire and False otherwise. sFlowID = the ID of the flow.
output	

AF_GetMakerType	

input	
output	an array with the following values: item[0] - action type -1 - not start new workflow action 0 - start new workflow (current user is the maker) 1 - start new workflow as a delegate maker (delegator is item[1]) 2 - copy form (current user is the maker) 3 - copy form as delegate maker (delegator is item[1]) item[1] - delegator UserID (makes sense if item[0] is 1 or 3)

AF_GetActivityName	
input	sProcessID = the ID of the activity
output	the name of the activity translated to the specific language preference of the current user.

AF_GetBubbleUpSteps	
input	
output	the number of bubble-up steps to get to the current user.

AF_SetBubbleUpTarget	
input	sUserID = the new route for the current workflow.
output	True if successful.

AF_SetTargetUser	
input	sUserID = new target userID sLinkID = the active output link ID
output	

AF_GetTargetUser	
input	sLinkID = the outgoing link ID

output	array with the following values. item[0] - target user type -1 - error (e.g. wrong link ID) 0 - the form hasn't been sent to anyone (e.g.final approval) 1 - bubble-up or match candidate routing (target user is a userID) 2 - group routing, target user is list of userIDs separated by TAB 3 - pool 4 - delegate maker item[1] - '' (item[0] = 0 or 3) userID (item[0] = 1 or 4) list of userIDs (item[0] = 2 - group routing)

AF_UserIsWorkflowMaker	
input	sUserID = the userID
output	True if the given user is the maker of the current workflow.

AF_IsUserInGroup	

input	sUserID = the userID
	sGroupName = the group name

output	True if the given user belongs to the given group. Note: if the sUserID is blank, the current user will be used.