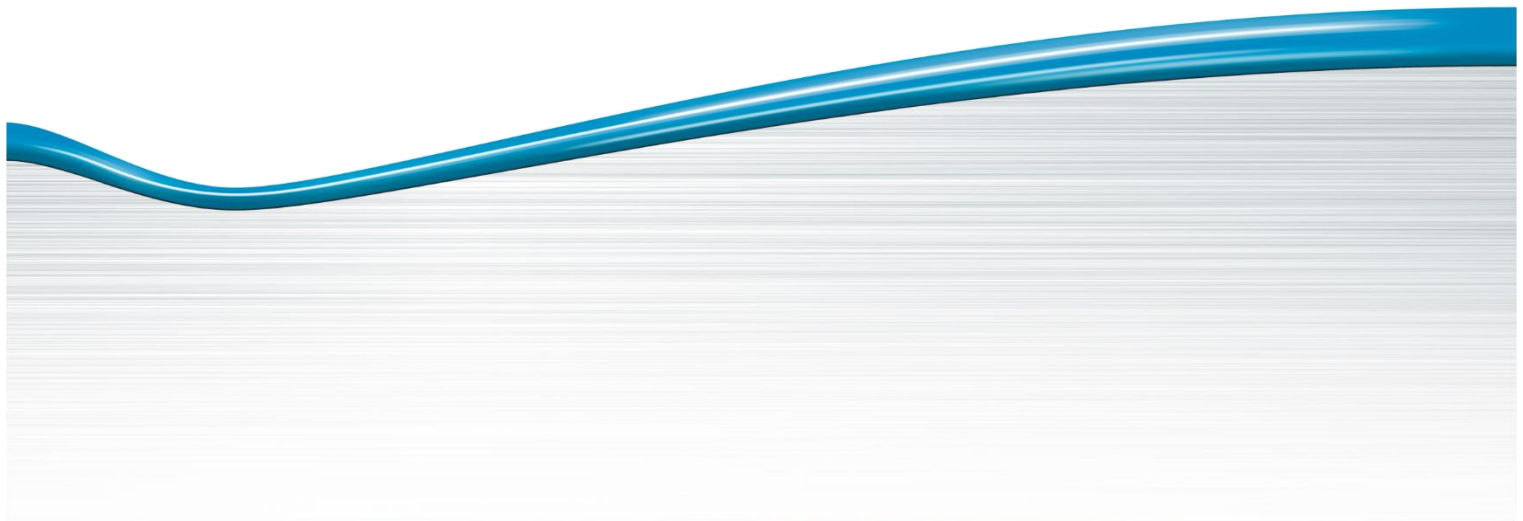


enaio[®]

Software Documentation
enaio[®] System Handbook WMS

Version 8.10



All software products and all related programs and functions are registered and/or used trademarks of OPTIMAL SYSTEMS GmbH, Berlin or one of its companies. They may only be used with a valid license agreement. The software and its associated documentation are protected by German and international copyright law. Unauthorized duplication and sales is plagiarism and subject to criminal prosecution. All rights reserved, including reproduction, transfer, translation, and storing on any media. Any names of companies and persons appearing in examples (screenshots) in preconfigured test scenarios or demo presentations are purely fictitious. Any resemblance to existing companies or persons is purely coincidental and unintentional.

Copyright 1992 – 2014 by OPTIMAL SYSTEMS GmbH
 Cicerostraße 26
 D-10709 Berlin

09/24/2014
Version 8.10

Contents

Foreword	5
Introduction	6
Basics	7
General	7
Process Models	7
Activities	7
Transitions	8
Roles	8
Splits and Joins	9
Loops	9
Variables, Applications and Parameters	10
Saving Process Models	12
Model Administration	12
Organizational Structure	13
Architecture	15
Process Design	16
General	16
Modeling Business Processes	17
Visualizing the Organizational Structure	17
Organization Explorer	19
Designing Workflow Forms	21
Visualizing a Model	22
Model Explorer	22
Model Editor	23
Extended Process Modeling	36
Loops	36
More Complex Loops	41
Multi-Instance Activities	47
Ad Hoc Activities	48
Events	49
Examples for Event Codes	52
Setting up the Log	57
Defining the Log Variable Type	58
Creating Log Events	58
Integrating Digital Signature and Authorizing Steps	59
Model Activation	59
Import and Export of Workflow Projects	60
Export	61
Import	61
Standard Ad Hoc Workflow And Taskflow	62
Standard Ad Hoc Workflow	63
Taskflow	70
Operation and Administration	72
Licensing and Rights	72
Workflow Components in enaio® client	73
Workflow Area	73
Settings	76

Substitution.....	76
Workflow Administrator	77
Process and Activity Status	78
Process Properties.....	79
Organization	81
Server and Database Properties	81
Statistics Reports.....	81
Workflow Simulation.....	84
Automated Starting of Workflow Processes	87
Integration Using the Import Wizard	87
Workflow Script Component	87
Microsoft Outlook Add-In	88
Workflow Examples	92
Leave Workflow.....	92
Order Workflow	92
Appendix	96
Analysis Sheet for Business Processes	96
Programming Reference	98
Environment	98
Standard VB Script Functions	98
Active Data Object.....	98
File System Object	99
Interface Objects.....	99
Report Configuration XML	100
General	100
Process Overview Report Configuration.....	100
Detailed Process Report Configuration	101
Report XML.....	102
Process Overview Report	103
Detailed Process Report	106

Foreword

This system handbook is aimed at all those who require information on process and data flows, interdependencies, structures, and technologies of the enaio® document management and workflow system beyond basic operation. It contains detailed information on the setup, configuration, and maintenance of the system.

We have tried to gather and systemize all information which is required to understand the structure, system architecture and interdependencies between the system components. Consequently, you should be able to get a better understanding of certain processes and to become aware of the idea, the advantages and disadvantages of specific settings, configurations and installations.

We have tried to avoid repeating information which is already contained in the user manuals as far as possible without causing misunderstandings. In some cases you will find references to further descriptions which contain information on parts of the DMS, e.g. database description, application server description, installation description etc.

This handbook will illustrate the system coherences during specific processes and describe the nominal condition during and after certain actions. You will be able to comprehend whether a certain action was successfully completed and where errors might have occurred. This system handbook will also assist you in resolving problems and recognizing errors.

This handbook does not contain explanations on algorithms which are technically realized for processes.

The handbook is divided into two parts. The first part deals with the document management system, and this second part with the workflow management system.

Introduction

The workflow component supplements the product spectrum of enaio® with a process-oriented component. It allows the user to map and control business processes electronically. The close link to the document management system offers significant optimization potential for the editing of business processes. Documents are available to authorized users during all processing steps and are automatically forwarded to the next editor after they are edited according to the specified process model.

enaio® workflow is a component integrated in the system and incorporates the proven advantages of enaio®:

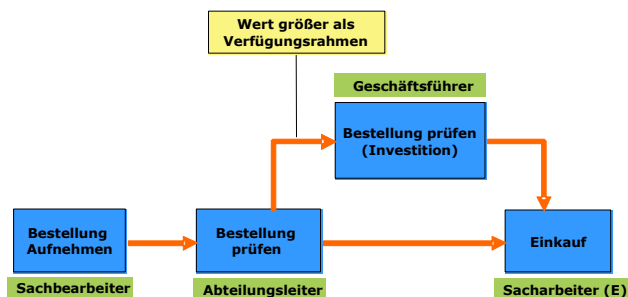
- Flexibility
- Configurability
- Stability

The administrative tools contained in the system allow business processes to be implemented quickly and safely, and administered and monitored during continuing operation.

This document describes the functionality of the workflow component and its integration into the system. Basics of the workflow management system will be explained in the following. In addition, the design of processes and organizational structures are examined, tools are presented and the operation of the workflow management system described.

Example

A company wants to support an order process with the use of a digital workflow system. The department manager approves purchases and in case an order surpasses a value of EUR 400 the chief executive officer will approve the order before the actual purchase can take place.



Of course, it is possible to map much more complex processes in enaio® Workflow; a selection of available scenarios is listed in the appendix.

Basics

General

A workflow management system (WFMS) is used to digitally visualize business processes within a company structure. The system can automatically forward processing steps to the assigned workflow participant and supports their processing.

In order to run a workflow management system it is necessary to visualize the users participating in the business process (e.g. employees in charge) and to model the required process (e.g. for leave application or invoice receipt) inside the system.

The participants of a workflow system are outlined in an **organizational structure**. **Process design** means to depict business processes in the WFMS as process models.

In order to understand modeling of business processes, each component of a digitally visualized process is outlined. The used terms are equally used in all workflow management systems which only differ in their appearance.

Process Models

A process model is an abstract depiction of a business process which is provided for a workflow management system in an interpretable form. Models depict a specific business case, processes depict the specific process. Hence, processes are instances of the process model.

The model design of enaio® workflow is based on the specification of the Workflow Management Coalition (WFMC), in particular the XML Process Definition Language. Beyond these specifications, optional attributes related to linked document management and content management systems have been added. Furthermore, not all of the interfaces required in the specification are implemented as they are not needed for regular system use.

Activities

A process model, e.g. the model of a purchase process consists of several single work steps which have each to be performed in order to complete the process. These work steps are called activities.

An activity is a delimited and related group of tasks that a user carries out in a step (e.g. confirmation of an order) within a process supported by an application (e.g. enaio® client).

Each process model has a start and end activity. These do not contain real work steps, but control the workflow process. Furthermore, other control activities can be used within a process model, which is discussed in the chapter on process design in greater detail.

In our example of a purchase order workflow the process model consists of the following activities:

- Start activity
- Take purchase order
- Review purchase order
- Review purchase order (investment goods)
- Purchase
- End activity

Multi-Instance Activities

Activities are always performed by exactly one participant. Multi-instance activities can be executed by more than one participant at the same time. Thus, this type of activity is in particular suited for polls and acknowledgments.

The transition from a multi-instance activity to the next activity is taken as soon as all participants have finished the activity by forwarding it. Alternatively, events can activate transitions.

Use local variables to administer data within the single instances of a multi-instance activity. Events merge these data.

Ad Hoc Activities

Ad hoc activities are areas in which activities without predefined transitions are integrated. For these activities, users deploy circulation slips in enaio® client to define the activity sequence and the user executing the activities. Circulation slips may define further processes and be designed in a way that the current user can edit and extend them.

Transitions

After defining that actions of a process can be split into a number of activities, you must determine in which order and under which conditions activities are performed. This is important because the workflow system is used to define activities for participants and assign them appropriately. Connections within a workflow process are separately flagged and called transitions.

A transition is a connection between two activities. A transition always has exactly one start activity and one end activity. Additionally, a condition can be specified under which the transition is realized.

A conditional transition can be illustrated using the above described purchase order workflow. The order is supposed to be forwarded from the department manager to the chief executive officer in case the order value exceeds EUR 400. In this case the way from the department manager to the chief executive officer is the transition, the expression (> EUR 400) is the condition. Transitions in the process model are the connections between activities.

Start activity	End activity	Condition
Start activity	Take purchase order	None
Take purchase order	Review purchase order	None
Review purchase order	Review purchase order (investment goods)	Value >= EUR 400
Review purchase order	Purchase	Value < EUR 400
Review purchase order (investment goods)	Purchase	None
Purchase	End activity	None

Actually, in a real life process further transitions are required, e.g. if the approval is not given and the process is returned to the person in charge. To keep it simple, these transitions are not discussed in greater detail.

Roles

So far we have described the basic structure of a process model. To run the process correctly in the WFMS, one must specify who will edit each activity.

The user is usually a person, but may also be a group of users, or a system component that runs automatically. All users, no matter if they are people or automated components, are simply referred to as **participants**. In our example, persons in charge, department manager, chief executive officer, and person in charge (purchase) are participants in the process model. In the context of an activity, participants are referred to as **editors**.

Since more than one participant can be responsible for editing a specific process step it is useful to group them to **roles**.

A role is a logical group of participants which have equal rights but cannot perform a specific number of work steps simultaneously. Role assignment of activities is part of the process model.

The example may involve more than one person in charge and all can equally process approved purchase orders. The 'Purchase' activity is designed to be edited by one person at a time. The person in charge picks a purchase order from the pool of assigned orders and processes it entirely. The activity can be performed equally (since each person in charge can perform the same task) but not simultaneously, since an order must be completed by the person which it has been assigned to.

Multiple users can be assigned to a role (Mr. Smith and Mrs. Miller are persons in charge), but one user can be part of more than one role (Mr. Smith is a person in charge and responsible for health and safety at work).

The example includes the following assignments between activities and roles.

Activity	Role
----------	------

Take purchase order	Person in charge (if the department manager and the chief executive office can trigger orders, they will be listed here as well)
Review purchase order	Department manager
Review purchase order (investment goods)	Chief executive officer
Purchase	Person in charge (purchase)

As mentioned above, these assignments are part of the process model.

Taking over an activity in a process by a specific participant within a role is called **personalization**. Once an activity has been personalized, it will disappear from the visible pool of activities for participants of the same role.

Besides having participants assigned to activities, other responsibilities may be assigned to respective roles. Specify the roles being authorized to start model processes and the roles monitoring running processes without being the actual editor of the current process step.

Roles can be assigned to people and process steps using server scripts. The roles saved in enaio® editor-for-workflow can be used, or completely new process roles can be created by scripts. Process roles have unique IDs. When a process step is created, objects of a process role can be assigned to the process step as participants. Future modifications of the process role do not affect the visibility of running process steps in the inboxes. Process roles will be displayed within processes in the workflow-recipient add-on and can be assigned this way.

Splits and Joins

In workflow processes in which branches are deployed, special activity properties allow branching and merging of single threads within the process model.

Splits

A split is a point at which at least two existing transitions are defined for an activity: the process splits into two different processing threads after the activity was processed. This applies to existing conditions with set conditions or to following activities that will be processed in parallel.

You can define splits as XOR or AND splits. XOR splits execute exactly one exiting transition for which the condition is fulfilled. AND splits execute all existing conditions for which the condition is fulfilled.

Joins

Joins are used to merge multiple processing threads. According to the type, a join can be executed when all incoming transitions or at least one transition was taken. These types are called AND joins (when it could be determined for all transitions if they were taken or not) or XOR joins (when exactly one out of multiple transitions was taken).

An AND transition is used to parallel process independent threads of a workflow process. For example, poll processes include different participants that accept or reject a process independently of each other. When merging the processing threads, the AND join indicates that the process cannot be continued until all threads have been fully processed.

XOR transitions are used to determine the processing thread to be taken according to the value of a variable. After processing, a XOR transition merges the threads, whereas it was defined in the process model that the process will continue as soon as one transition is completed.

By default splits and joins are set to AND to keep the design work effort at a minimum.

Loops

In real-case processes it is often required to repeat parts of a process until a specific state is reached. E.g. an ordering process is forwarded between the requesting employee and the responsible department manager until an explicit approval. This may be necessary in case of further questions during an ordering process or not all information is available.

That is why loops are available in process models. Loops are special activities that have a specific type (while, repeat until) and a loop condition. Within a loop, one or more activities can be set up that will be performed as long as the loop condition is fulfilled.

Variables, Applications and Parameters

The term 'condition' has already been introduced above. Let's discuss how these conditions are interpreted. **Variables** defined in the model are used for this purpose. They are available at all time during process execution and can be read and modified by the WFMS. In particular, they are used to define conditions. Variables have a predefined data type and can be preset with values at process start. All types already known from the DMS (integer, real, string, date etc.) are available. If required, custom data types can be defined by the designer to administer structured data.

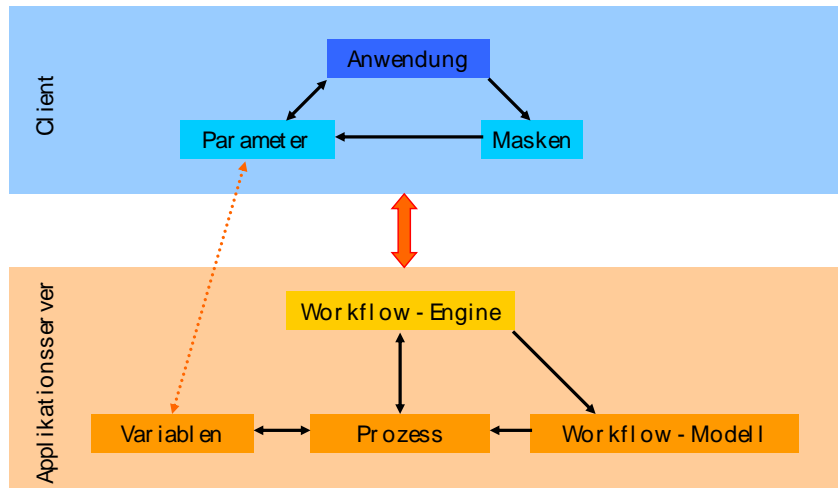
In the following example a condition has been specified for a transition which will be taken if the purchase order value exceeds 400 €. The variable 'value' is introduced in the process model. The person in charge taking the order fills in the value of the purchase order, i.e. the variable value. Now the variable can be evaluated in the transition condition.

Until now, we have been talking in general terms about a workflow management system. In actuality, this type of system consists of multiple parts that are at least logically separate. Most systems run a workflow engine, i.e. software that monitors the start and processing of the process. Usually the workflow engine is part of an application server; for smaller department solutions, workflow engines may also be integrated directly into the end-user application. Server solutions are more stable, powerful, and scalable. enaio® Workflow uses a server-based workflow engine.

The workflow engine controls the processing of processes of which multiple can be simultaneously active. It monitors when an activity must be started, stopped and for whom the next-following activities within a process are intended. In this context it is important to understand that the engine only controls processes, whereas activities are executed by **applications** that communicate with the engine. Applications are used to process activities whereby it is unimportant to the workflow engine if it is an interactive application that requires the user to edit data in a form, or another software instance which automatically executes operations based on the data provided. The application must be able to exchange data with the workflow engine in order to participate in a running process. The most important kind of data which can be manipulated by applications is process variables. These must be handed over to the application and returned to the workflow engine after processing. This requires **parameters** to be defined for the applications. Parameters are assigned to variables in the model which creates a link between the workflow engine and the application.

Like all applications in the enaio® product family, enaio® Workflow has a client-server architecture. Activities processed by the user usually take place in enaio® client. In this case, enaio® client takes on the role of an application that exchanges data for processing with the workflow engine.

This is visualized in the following illustration.



As mentioned above, it is not essential to process activities in an interactive client application. An application that automatically processes data along with other sub-systems may also execute activities. Here also variables are depicted on application parameters.

Application parameters can be of a different type depending on the direction of the variable interaction. The following types are defined:

Parameter type	Meaning	Example
In	Input parameters – the parameters are transferred by the engine to the application but cannot be returned.	<ul style="list-style-type: none"> - Static texts which are displayed as notes on the input form. - Runtime-parameters for applications – these do not have any further effect on variables.

Out	Output parameters – the parameters are transferred from the application to the process without having been transferred previously.	<ul style="list-style-type: none"> - Input variables which are transferred using out-parameters by the application on process launch. - Initial initialization of variables
In/Out	Parameters that are transferred to an activity for editing.	<ul style="list-style-type: none"> - Parameters which are linked with input fields and are intended to be changed. - Parameters for validation and editing in an activity

enaio® Workflow is the most frequently used application of enaio® client. That is why the configuration options have been extended for added convenience. Interaction between the workflow engine and the user is via input forms, which you are already familiar with from the DMS. These forms can be created graphically and then linked to the process model. The data fields on input forms act as application parameters and can be assigned to variables.

For the purchase order workflow exemplified here, an input form would be set up for each activity into which users can enter his data. One form may also be used for more than one activity.

The screenshot displays the enaio® Workflow application interface. The main window, titled 'Bestellung*', contains a form for entering purchase order data. The form includes fields for 'Artikel' (Article), 'Kategorie' (Category), 'Wert' (Value), and 'Bemerkung' (Remark). There are also checkboxes for 'genehmigt Abteilungsleiter' (approved by department manager) and 'genehmigt Geschäftsführer' (approved by CEO). The 'Eigenschaftsfenster' (Properties window) is open on the right, showing the 'Bestellung' object with its properties: 'Datenbank' (Database) set to 'Bestellung', 'Position/Größe' (Position/Size) with 'Breite' (Width) 189 and 'Höhe' (Height) 240, and 'Flags' (Flags) set to 'Sonstiges' (Other).

In total, four applications (one for each activity) would be set up in the process model and linked to the specified forms. The applications can be named, whereby it is recommended to name them after the activities.

The variable 'value' which is important for the process flow was already named above, as it dictates whether the process is forwarded to the chief executive officer for approval, i.e. a transition must be taken to the activity 'Validation chief executive officer'. Since input data from the order transaction must be visible at all stages, variables which will be manipulated by the activity need to be deposited in the model. In this example the following variables are defined:

Variable	Data type	Meaning
Article	String	Item label
Category	String	Category of the item (e.g. office supplies, IT...)
Value	Integer	Value in €
approved department manager	Integer	Specifies, if the purchase order must be approved by the department manager (check box)

approved chief executive officer	Integer	Specifies, if the purchase order must be approved by the chief executive officer (check box)
Reviewer's comment	String	Text for annotations

The variables can have any name, but it is useful to name them in a similar way as on the form.

Parameters which are linked to process variables are specified for each application. Specify with the parameter type if values at a particular activity can be processed. In this example it makes sense to make the parameters 'approved department manager' and 'approved chief executive officer' write-protected, i.e. to define them as In-parameters.

In enaio® Workflow, the data fields on forms are set as application parameters by default.

Process Interface Variables

Special features of the variable administration are interface variables of processes. They depict the process interface. Thus, the variables can be seen as input/output parameters of the entire process. This applies to the following two scenarios:

- Starting a process with variable transfer
- Exiting a process and variable transfer to the executing instance

Interface variables are mainly used for communication between multiple independent processes. This allows transferring results of a specific process in the form of variables to the succeeding process. Hence, models can be created and operated independently of each other.

Based on the example, it might be possible that after the activity 'Purchase' and the intended process end, specific variables are transferred to another process in the enterprise resource planning system. Interface variables are furthermore used in the incoming mail scenario to transfer data from enaio® capture to the workflow system.

Saving Process Models

As mentioned above, a process model is described in a meta language and saved in the system. Various standards exist for describing process models. In enaio® Workflow, the XML Process Definition Language of the Workflow Management Coalition is used. As the name suggests, all information belonging to the model is expressed in XML. This allows quick access, structured display, and a high degree of compatibility.

The specifics of enaio® Workflow are available as extended attributes in accordance with the standard. In particular, these include links to workflow forms and attributes to attach digital signatures and to depict workflow files.

Two options are available to exchange process models which are described in greater detail in the chapter on process design and on operation and administration. One option is the saving of the pure XML-model without resolved references to forms; the second option is the organizational structure and the import-export functionality.

Model Administration

During operation, process models often perform different processing stages. These consist of:

- Analysis of the organization and the business processes
- Model design
- Operational deployment
- Release for running operation
- Use for already started processes

An integrated model administration is required for the different processing stages to be accompanied correctly by the workflow management system. It is used to differentiate between separate projects and allows versioning of the process models within projects.

In enaio® Workflow, process models are organized into workflow projects and workflow families.

A **workflow project** consists of one or more sub-projects or workflow families and summarizes thematically coherent process model groups.

A **workflow family** consists of one or more single process models. Processes within a family can have different processing stages, but there can only be one active process model. Workflow families allow the versioning of models whereby the user administers the versions.

Below is a list of the various processing stages of a process model in enaio® Workflow.

Processing stage	Description
Locked for editing	The model is currently being created or modified by a user. Other users have no write access to the model. The model is checked out of the system.
Available for editing	The editing of the model was finished with a preliminary result. The model has been checked in to the system. It can now be edited again or used for current operation.
In use	The model is in regular operation and loaded by the workflow engine. Usually, this status is set while switching between model versions when already started processes are supposed to complete with their current version, but new processes of this model must not be started.
Active	The model is in regular operation. New processes of this model can be started by authorized users.

Organizational Structure

Every company with a larger number of employees maintains structures to organize, distribute, and process business tasks. These structures can be divided into organizational groups of employees, e.g. by location, department etc. and also by scope of duties, e.g. production, sales, development. In many cases both structures co-exist.

Important information for the organization of business processes can be derived from these structures. A pivotal question is who is responsible for the processing of which scope of duties. This is depicted by the above introduced term **role** in a workflow management system.

Roles are either derived from the hierarchical structures of a company or from the tasks which the users mainly process. Hence, the following roles can be defined:

Organizational relation

Employee of the production department

Chief executive officer

Employee at the location Berlin

Task-orientated supply

Submitter of an order

Sales order validation

Application editing

As already mentioned above, a real user can be assigned to multiple roles and one role can have multiple members. If you imagine the number of possible business processes and the resulting number of activities and roles, it becomes vital to use the system in order to simplify the depiction of structures.

Usually, in larger companies employee structures are administered by a directory service (e.g. LDAP) in which the organizational membership as well as roles and attributes of the employees are depicted.

The workflow management system is used to create the possibility to transfer these structures into the system which then allows the assignment of employees to roles. This allows a user to have exactly the tasks or activities assigned which are destined in the process model for his role.

Hierarchical and task-related structures in a company are combined in a **organizational structure**. Since companies mostly have different structures, it is necessary to allow company-related structuring, i.e. to classify groups of similar elements and to depict them with predefined rules.

An organizational structure consists of organizational classes, structure elements, and the assignment of elements among each other. Elements of an organizational structure are called **organizational units**. Examples for organizational units are e.g. production department, secretary's office or employee Mr. Smith.

Organizational classes depict the structure of a specific type of organizational units. The depiction is carried out by specific attributes which are common to all organizational units of this type. Examples for organizational classes are: country, department, team, and person. Organizational classes create the meta-structure of an organizational structure.

Specific **attributes** which specify single elements in detail can be depicted for all organizational classes. Each organizational class has a defined attribute, the name of the element. It is used to specify each element of a class. Attributes can be assigned during the design of the organizational structure. Attributes are important for the operation of workflow management systems, since the values are accessed during the editing of processes which has an effect on the

progress of the process. In particular, the access of attributes of the participating users may be useful when defining conditions in transitions.

The following table lists examples of organizational classes and associated attributes. The listed organizational classes are already defined in enaio® Workflow, but can be added to as desired.

Organizational classes	Attribute
Country	Continent Sales territory Official language
Organization	Founding year Legal form Charitable disposition
Department	Producing Location Cost center area
Person	First name Last name Date of birth Recruitment date Disposition limit

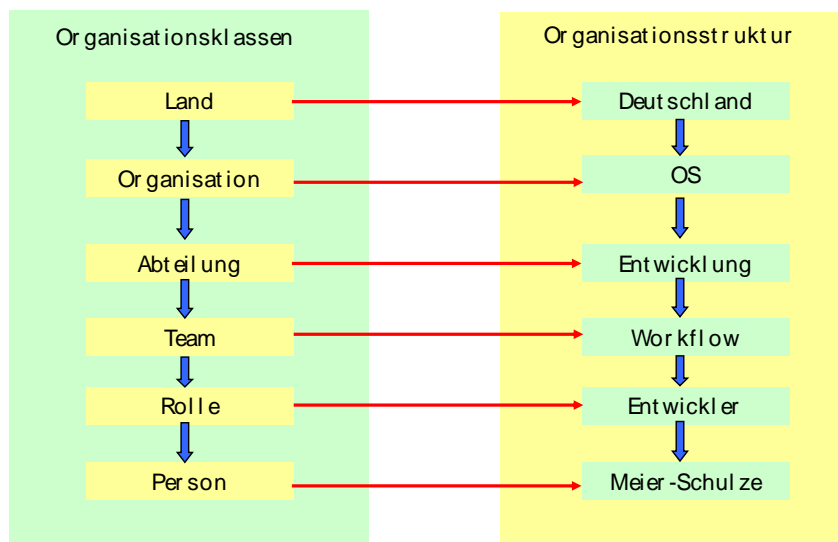
The role is a special organizational class which is predefined in the workflow management system.

Furthermore, organizational classes can be used to specify in which form single organizational units are assigned to each other, i.e. which kinds of hierarchies are possible. For example, you can thus specify that a person must always be assigned to a particular department, but never directly to a country.

Organizational units are instances of organizational classes, i.e. their special characteristics. Each element inherits the attribute's definition of the organizational class. However, attribute values are specified for each element. For example, the cost center area for the production department can be provided with the value 3124-3152 or the date of birth of person Mr. Smith with the value 12.03.1962.

Organizational units can be arranged hierarchically, and multiple assignments to superior organizational units are available. This may be required for persons being members of more than one department due to organizational reasons.

The relation between organizational classes and organizational units is illustrated by the following image.



Organizational structures are usually depicted according to the X.500 specification in workflow management systems and directory services. This requires organizational units to be depicted as related, oriented, non-cyclic graphs. The term X.500 tree is often used in this context, although the graph is not a tree according to the graph theory due to available multiple assignments to parent instances.

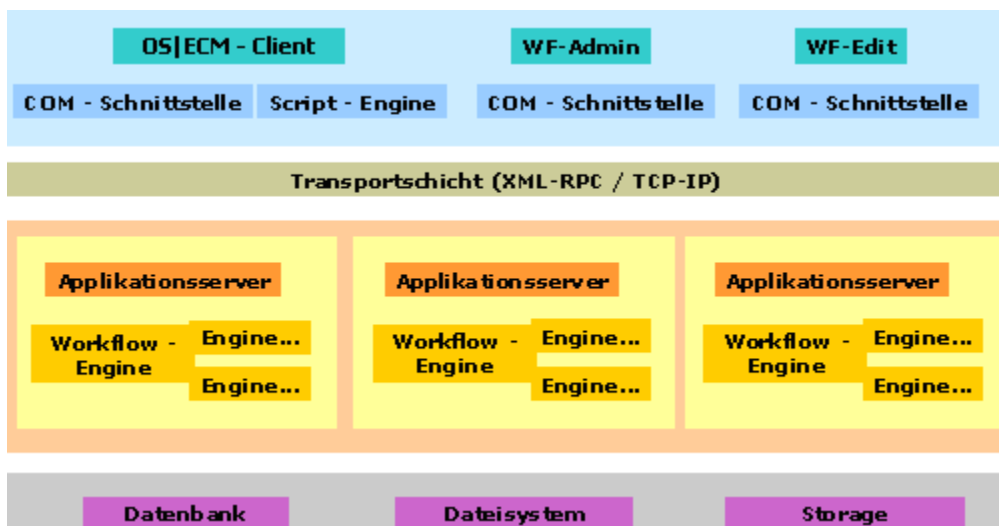
Architecture

The workflow engine is an integrated part of the enaio® application server. It is loaded as an executor (oxjobwfm.dll). All communication with the workflow clients is handled by server jobs. This means that applications cannot directly access processes, models, or other data structures within the workflow management system. The workflow engine runs on the business layer in the 3-tier architecture model. This allows the workflow to be scaled in almost unlimited ways as load balancing is applied to application servers.

The following modules are used to integrate workflow jobs on the Client:

- oxwfcInt.dll: COM-library depicting the query interface of startable processes, contents of inboxes, editing of activities, administration jobs etc.
- oxwfedit.dll: COM-library depicting the interface for process design and for administration of the organizational structure
- oxwfjobc.dll: library depicting functions in the XML-RPC syntax, for workflow jobs execution, and for result processing.

The following image illustrates the architecture.



Only one job which encapsulates all function parameters and results in an XML structure is used for job communication with the workflow engine. Hence, third-party systems can be integrated on the transport layer by implementing a function call based on XML-RPC.

In addition to the job call that launches functions of the workflow engine, e.g. process start or model saving, the application server uses the notification channel to inform the client about changes. A new activity in a user's inbox would principally be such a change to be reported.

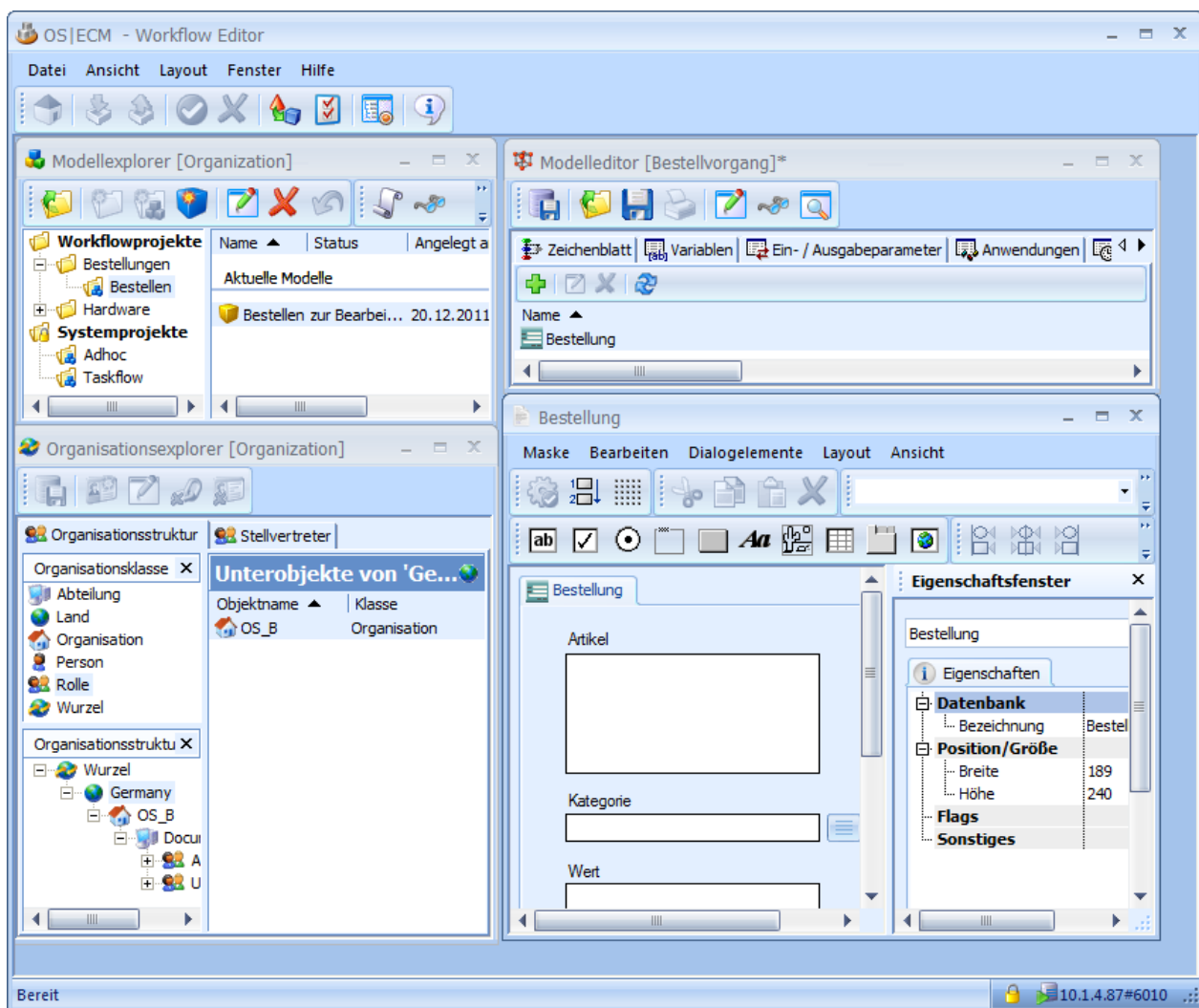
In this case, the application server sends a message to the client that will consequently refresh its inbox. As a result, system load is reduced as launched clients do not need to consistently query the inbox content from the workflow engine. Actually, the server only uses the active TCP connection to notify the client. No additional technical connection is established between the server and the client.

Process Design

General

This section contains a step-by-step description of how to design a business process with the enaio® Workflow tools. The theoretical principles discussed above are implemented with practical examples. Processes are usually designed in enaio® editor-for-workflow, which provides the following functionality:

- visualization of the organizational structure
- form designer for workflow forms
- process model design
- import and export of models and complete organizational structures



All the application functions described in this chapter relate to enaio® editor-for-workflow. Other programs that are relevant for the operation of enaio® Workflow are named in the 'Operation and Administration' section.

The applications are installed as part of the administrative components into the directory \clients\admin.

Modeling Business Processes

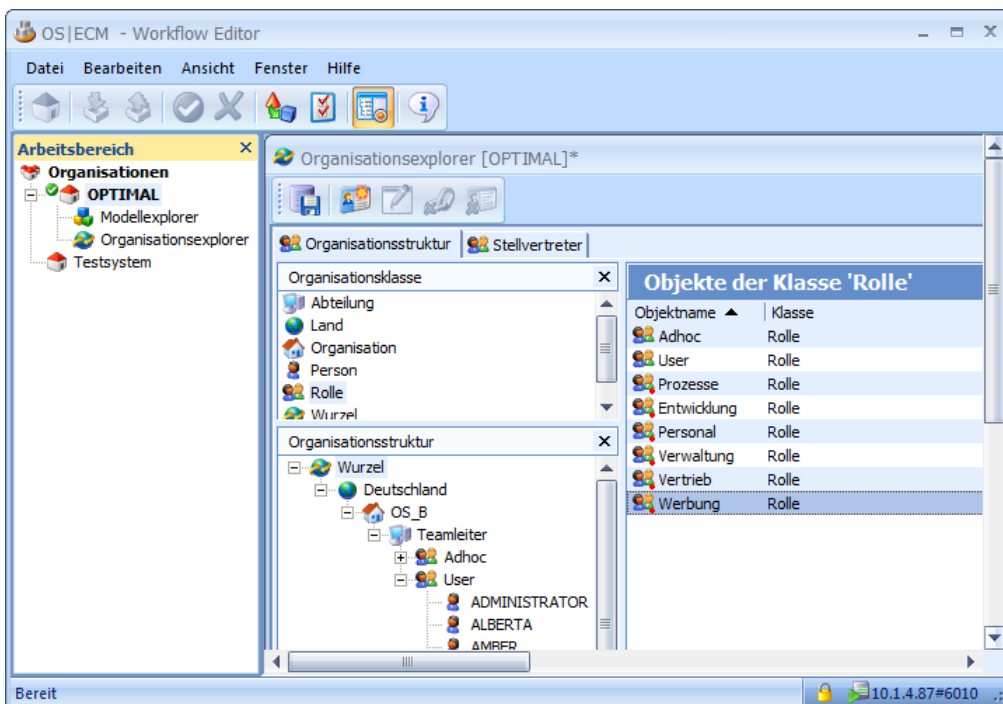
As mentioned above, an effective operation of a workflow management system requires detailed analysis of business processes before modeling. In doing so, the following points need to be clarified:

- Who participates in the business process?
- Which are the sub-steps the business process consist of?
- How can sub-steps be depicted as activities (with input forms)?
- Who is allowed to start business processes?
- Who is allowed to process individual activities?
- Who is allowed to monitor the progress of processes?
- Is there a connection to the DMS and how is it realized for individual activities?
- Which transitions exist between individual activities?
- Under what conditions are transitions valid?
- Which return points exist in case an activity has not been processed?
- What will happen to documents of the workflow file at the end of a process?
- Which activities will be executed automatically while processing activities?
- What deadlines must be met?

The appendix contains an analysis sheet for business processes which facilitates finding answers to these questions and consequently designing the process. It is recommended to perform an analysis based on this sheet before actually designing the process in the system.

Visualizing the Organizational Structure

Use enaio® editor-for-workflow to create multiple organizations with unique names in the system. Organizational classes, the organizational structure, and any number of workflow projects belong to each organization. Add or delete organizations with the context menu of the "Organizations" node.



One organization is 'active' at a time. Workflow models that are assigned to the active organization can be used only. The active organization is flagged with a green checkmark. Activate an organization through the context menu.

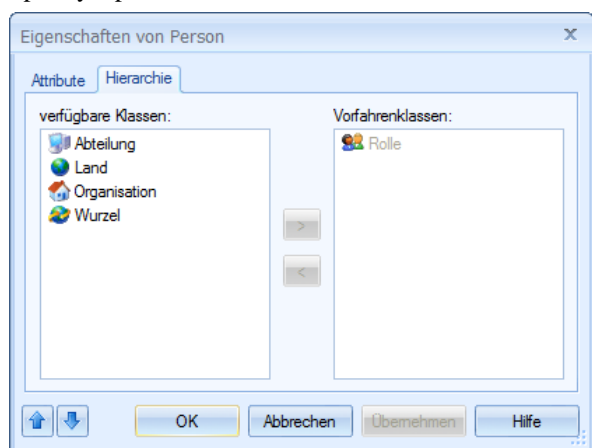
If an organization is deactivated by activating another one, all running processes of the organization are instantly stopped. Users will be notified in enaio® client and data in the 'Workflow' area will be updated.

Once a new organization has been created, organizational classes can be edited. Six organizational classes are already predefined by the system.

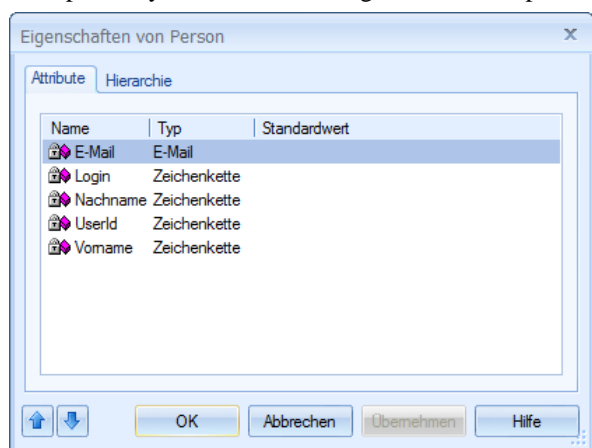
Organizational classes	
Root	
Country	
Organization	
Department	
Role	
Person	

If required, create further organizational classes through the context menu. You can enter a name when creating an organizational class.

Specify a parent class to determine where instances of the organizational class are available.



Organizational classes can be provided with predefined attributes. All instances of the organizational classes can allocate values to these attributes. These attributes are used to set specific properties to organizational units. Attributes examples may be: location for organizations, disposition limits for departments, or the date of birth for persons.



The organizational classes Role and Person are of particular importance. In the process model, roles and persons can be assigned as participants to activities. Furthermore, roles and persons can start processes and monitor its progress if the model was configured accordingly. To connect roles and real users, you must assign set up persons to corresponding roles and also a corresponding login name to persons. That is why the attributes Login and UserId are predefined for the organizational class Person. These attributes cannot be deleted. Add, edit and delete attributes through the context menu.

The following dialog is used to define the properties of the attribute and to preset a default value.

Neues Klassenattribut

Name: Titel

☒ Optional

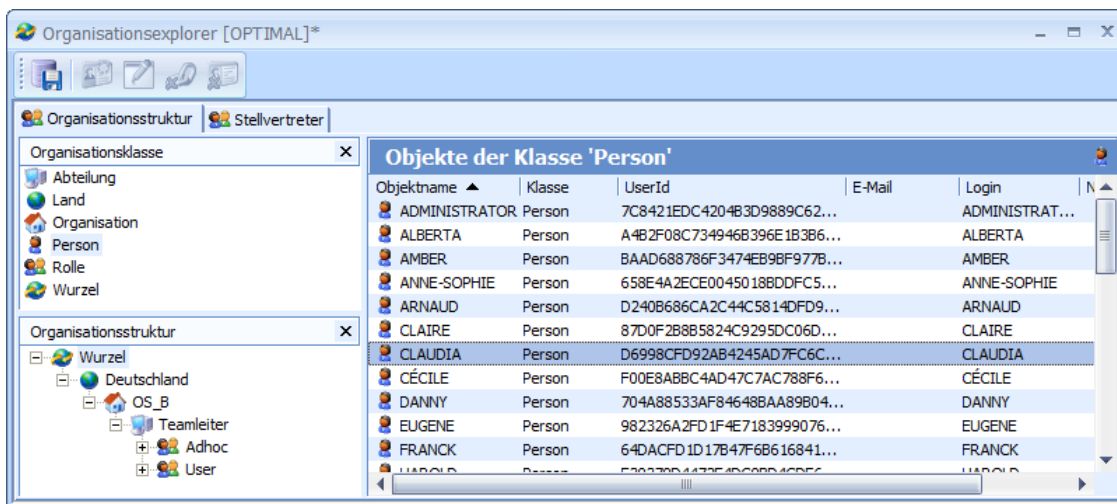
Typ: Textfeld

Standardwert:

OK Abbrechen

Organization Explorer

The organization explorer offers a highly structured view on relations of organizational units and its intuitive operability facilitates the creation of organizational structures.



The organizational structure is actually defined with the organization explorer. Herein, instances of respective organizational classes and subsequently the structure are defined. Two steps are required to define the organizational structure.

Creating Objects

Select an organizational class and create a new object using the context menu or the corresponding button. This object needs to have a name assigned and have its attributes preset which are defined in the organizational class.

As mentioned above, the attributes UserId and Login need to be filled during creation. This is done by assigning the person to a DMS user as described below. For other attributes it depends on whether the attribute was defined as optional in the organizational class.

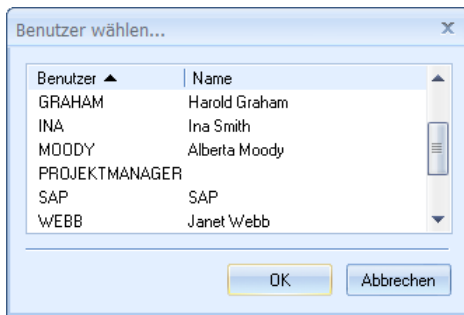
Eigenschaften von Jill

Attribute | Hierarchie

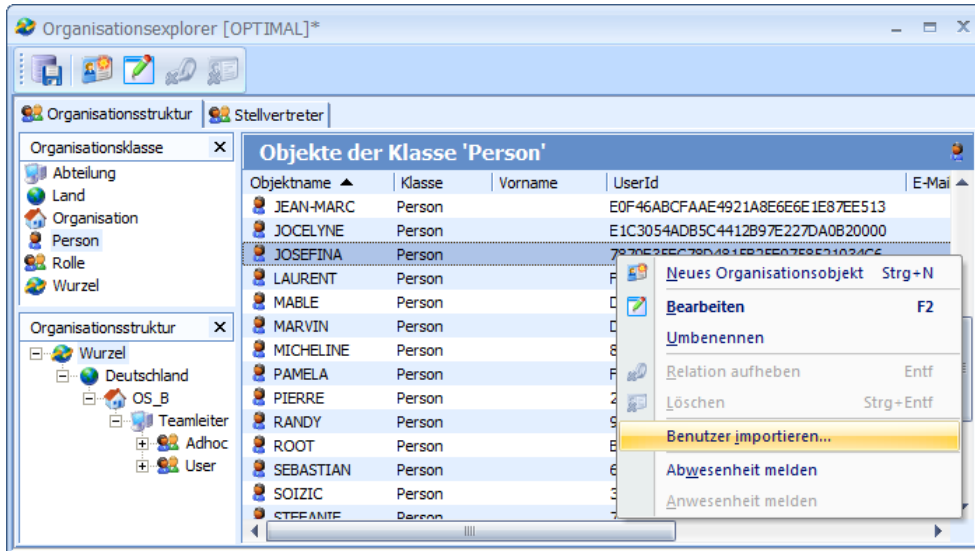
Name	Wert
E-Mail	jill@optimal-systems.de
Login	INA
Nachname	Schall
UserId	8C237171833E48FB870E5B95E87EAF5C
Vorname	Jill

OK Abbrechen Übernehmen Hilfe

Assign a DMS user through the selection dialog in which the login name is displayed.

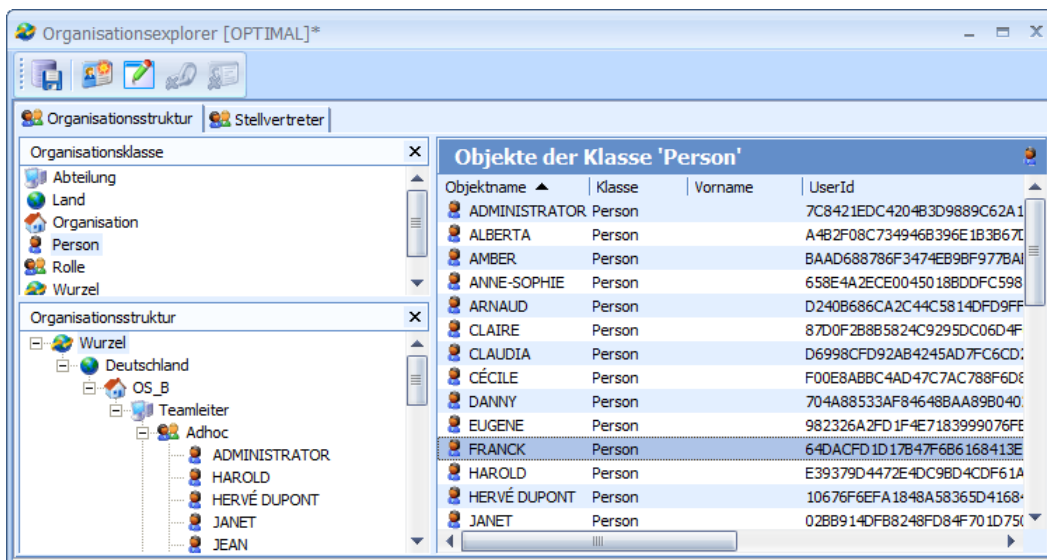


A person corresponding to already configured DMS users can be created automatically in the organization. To do so, use the **IMPORT USER** entry from the context menu in the object view of the organization explorer.



Assigning Objects to Organizational Units

After the objects have been created, they can be assigned to parent organizational units in the tree view. For this purpose select an organizational class to display existing objects in the list. These can then be assigned to organizational units by dragging and dropping. Objects still not assigned to any organizational unit are flagged with a red icon in the list.



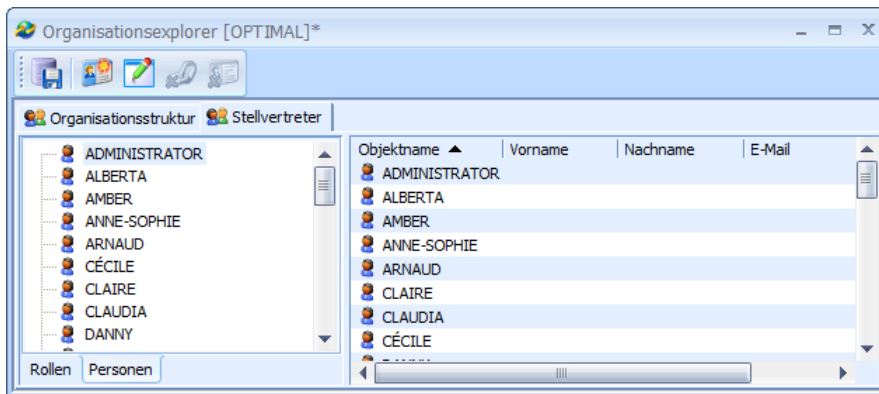
Since the organizational structure is not a tree but a non-cyclic, directed graph, objects may be at multiple places. To improve clarity the tree structure was chosen in the organization explorer. When dragging and dropping it is verified if the objects can be inserted into the organizational classes according to the specified hierarchy. A not supported assignment cannot be performed.

Substitution

Substitutes are also set up in the organization explorer. Persons and roles can be assigned as substitutes for other persons. Users indicate their presence or absence in enaio® client. If all participants of an activity are absent, the activities are displayed in the inbox of all substitutes. Activities that were already personalized are also forwarded.

Activities are not forwarded to the substitutes of substitutes.

The substitute administration can be opened on the **SUBSTITUTES** tab.

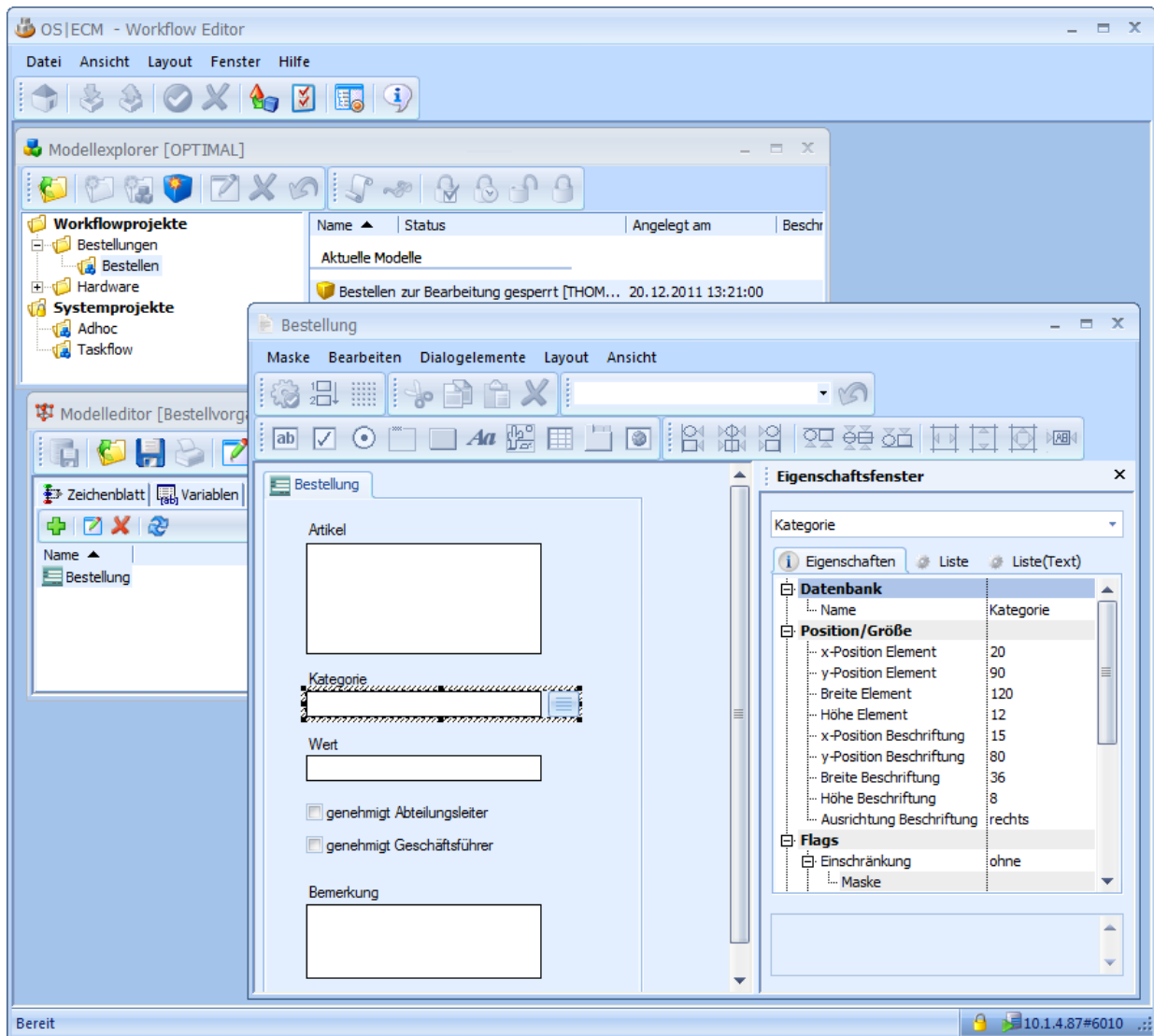


Persons can be assigned by dragging and dropping to all persons or roles from a list of available substitutes.

Substitutes also apply for follow-ups of absent users.

Designing Workflow Forms

Workflow forms are designed in enaio® editor for workflow in the same way as in enaio® editor. However, only properties are available which are not especially destined for the DMS.



Workflow forms are created for a process model (workflow model), but can simply be copied to and used in other models. The forms are later used as application interfaces for process design. A parameter must be assigned to each field on the form.

Forms contain fields; fields have functional properties which are specified in a properties dialog, and graphic properties which are designed with the graphic interface.

For workflow forms, size specifications apply:

- Forms can have a maximum height of 750 dialog units and a maximum width of 1000. If no values were specified, they will be displayed in enaio® client with a minimum height of 180 dialog units and a width of 220 dialog units. In order to assure complete visibility of elements, they have to be positioned completely on the form.

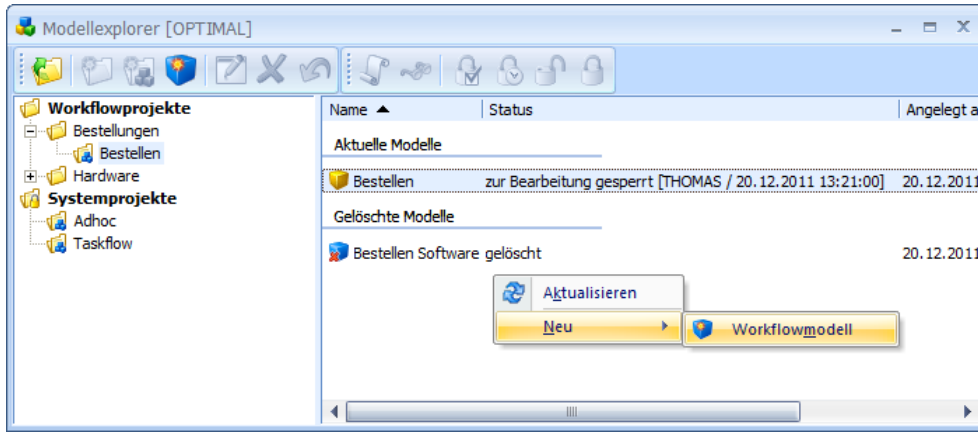
Visualizing a Model

A model is visualized in multiple steps which are described below. In any case, it is required to have an exact idea of the business process to be depicted.

Create workflow projects in the workspace of the workflow editor and assign workflow families to a workflow project. All data of a workflow family are edited in the model explorer.

Model Explorer

Models are administered within a workflow family. Create a new model in an existing workflow family using the context menu of the model explorer or the corresponding button.



The model has a name which is unique within its family.

After having created a new model, it is instantly opened in the model editor. Therein, the model's status is automatically set to 'locked for editing'. The user is also displayed. Only this user can edit or release this model. Other users can open the model as read-only or create and edit a copy though.


The model status can be changed with the buttons on the model explorer toolbar (see 'Model Administration').

Only one model of a workflow family can be set to 'active'.



Validate the model before activating it. The function checks if all necessary data has been entered. If this is not the case, a notification dialog will open.

A validation does not guarantee that the process is error-free.

Deleted models are transferred to the **DELETED MODELS** tab. Data have still not been deleted from the database. You can restore the model  or delete it permanently.

Use the model explorer to also open the event editor. You can create events with the event editor. An event is a VB script assigned to an activity in enaio® client that is launched automatically by an activity. For example, a VB script can perform a validity check or automatically complete data when forwarding (see page 49).

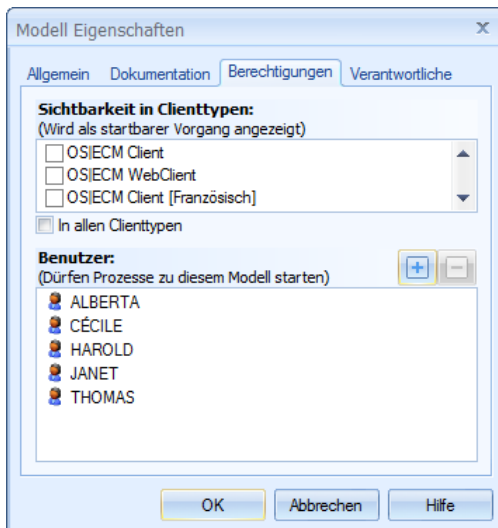
Model Editor

Double-click a model to open the model editor.

Seven tabs allow you to configure a model:

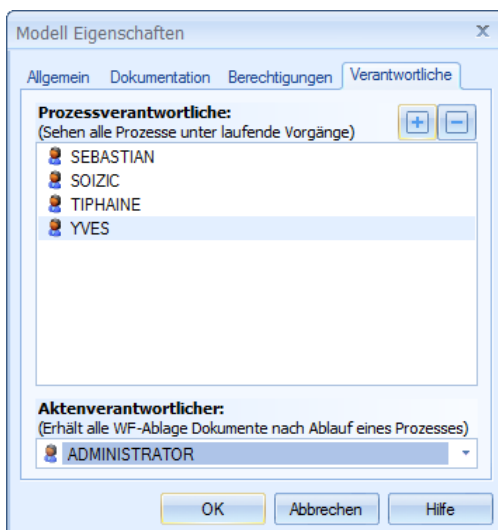
- **DRAWING PAGE**
Activities and transitions between activities are graphically defined on the drawing page. Depending on the way of working, you can either start with a graphic model draft or specify the variables and applications first.
- **VARIABLES**
Variables are defined. Activities will change the variable values.
- **INPUT/OUTPUT PARAMETERS**
Variables can be defined as input or output parameters. External processes can then access these variables.
- **APPLICATIONS**
Applications are workflow forms with fields. Applications are assigned to activities. You can determine which variable can be changed by a field in the workflow form within an activity.
- **DUNNING AND RETENTION PERIODS**
Dunning periods ensure that activities or sequences of activities are processed within a specific period of time. Retention periods are used to specify that activities or sequences of activities cannot be completed before the end of a certain period of time.
- **FORMS**
Forms allow data input in client applications.
- **EVENTS**
An event is a script (e.g. VB script) that is assigned to an occurrence in the process flow of a workflow process. For each model, you must additionally define which user is permitted to start a process of this model and from which clients this model's processes can be started at all.

Enter users and client types into the property dialog of the model on the **AUTHORIZATION** tab.



Specified process supervisors can observe the process progress by viewing model processes in the **Running processes** tab in enaio® client. In addition, a document supervisor can be set up. This person then receives documents which are still in the workflow tray after a process was finished.

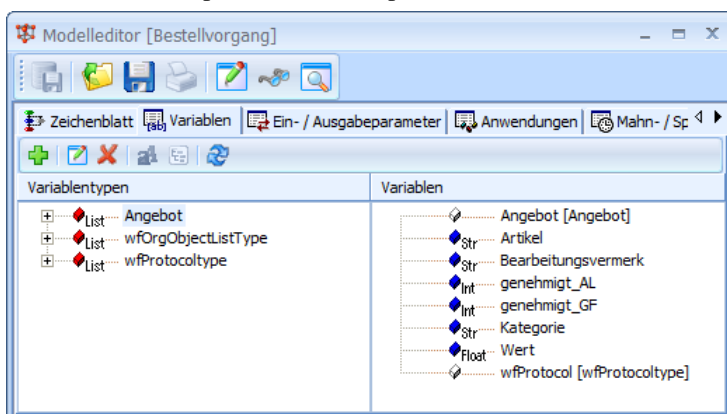
Supervisors are entered on the **SUPERVISORS** tab of the model's properties dialog.



Variables

Variables contain data which affect the flow during process execution and which can be modified by activities. Therefore, they have to be defined in the model first.

Variables are set up in the model explorer on the **VARIABLES** tab.



The variable `wfprotocol` is preset. This variable of the 'List (Record)' type is used for logging and is assigned to an application parameter and administered with scripts.

Variables are typed, i.e. the data format is primarily determined. Available are simple, composite and custom variable types.

Variable names must neither contain umlauts, nor spaces or other special characters.

Simple types

- String** contains a string based on the standard character set. The length limit is predefined by the underlying database system and refers to the maximum size of a BLOB field (size depends on the DBMS in use).
- Integer** stands for an integer ranging from -2^{31} to 2^{31}
- DateTime** gathers date/time according to the date format set in Windows
- Float** contains a floating-point number

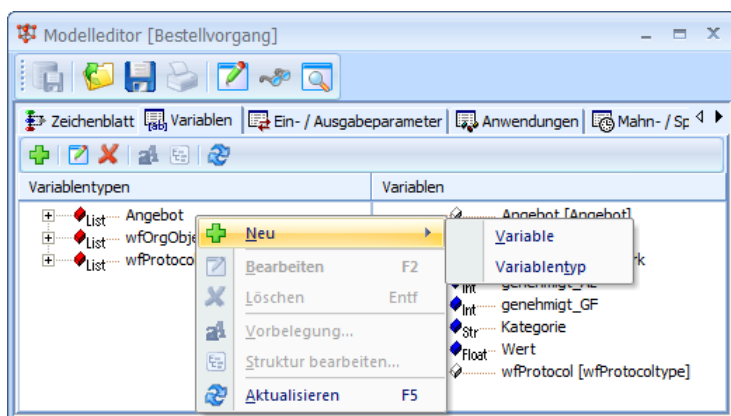
Composite types

- List** is a list of data of a simple, composite (except list) or custom type
- Record** represents a composite structure of multiple simple types

Custom types

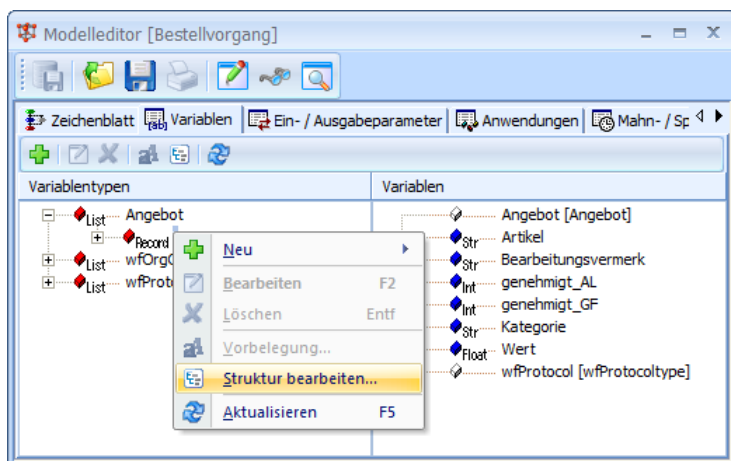
The system allows you to define complex data structures as variables. This may be necessary when not only values, but structures like tables or lists of different entries need to be transferred as variables. For example, logging is preset as a custom type as it consists of a list of records. These variable types can be defined by the user.

Variables are set up on the **VARIABLES** tab.



Select **NEW VARIABLE** from the context menu, enter a name, a description (optional), and a type.

If 'Record' was set as variable type, the structure can be modified using the **EDIT STRUCTURE** entry from the context menu. A dialog will open in which you can set up individual variables which will be combined to a record.

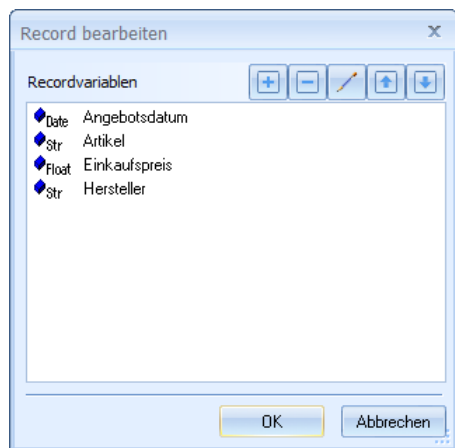


Setting up custom types

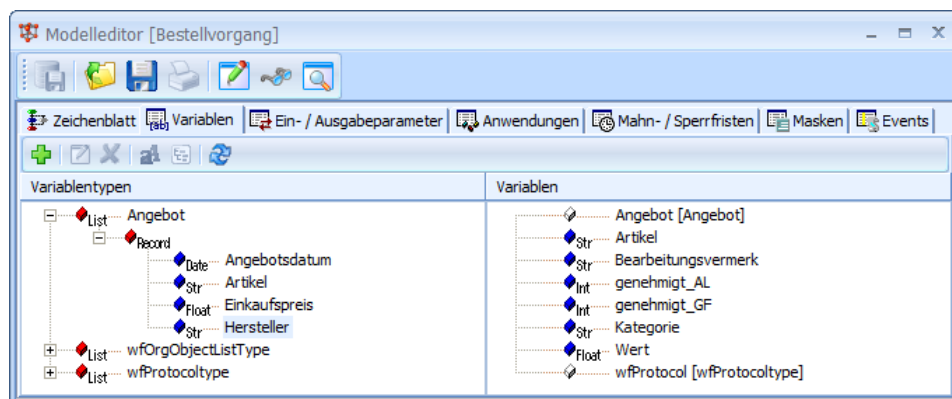
The example may contain a table in which the person in charge can enter offers for the purchase order. In such case, a record list would have to be defined. Each row will gather a record and all rows together represent the table. That way, the entire table can be saved in a single workflow variable.

To do so, create a new variable type 'Offers' of the type List (Records). This list will gather records with a structure that is defined in the following dialog. The structure will contain an offer date, an article description, the manufacturer, and an offer price. Hence, variables of an adequate type must be added to the record structure.

Element variables of a record are added in the same way as simple variables. Variables must have a unique name in the record and a simple type. It is not possible to nest records in other records of the same type.



After you have defined the new type, specify the corresponding variable by selecting the created record from the list of available types.



Input and Output Parameters

If you want to start processes with a preset value, interface variables must be defined. This may be necessary in the following cases:

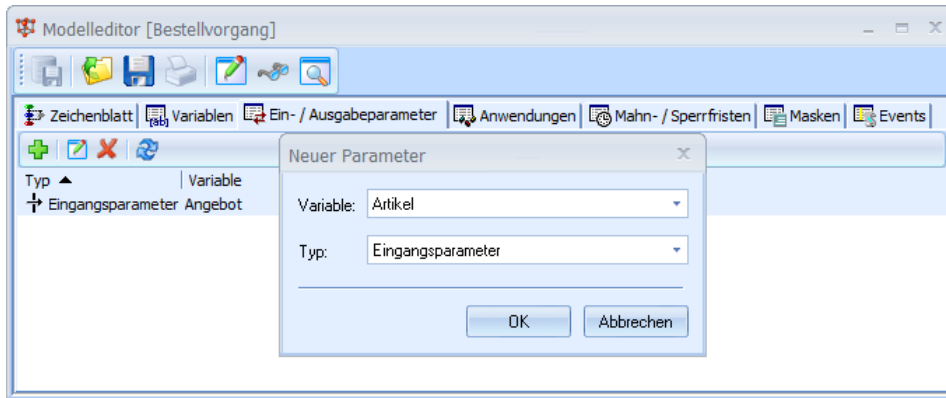
- Starting processes in enaio® capture using the workflow start component
- Starting processes in Microsoft Outlook with e-mail data transfer
- Linking multiple equal processes

Values for the interface mode of variables are available as follows:

- Input parameters:
the variable is transferred at process start. This happens when an external application starts a workflow process.
- Output parameters:
the variable is returned at process end.
- Input/output parameters:
the variable is transferred at process start and preset with values at process end.

Only variables which have been set up on the **VARIABLES** tab can become interface variables of workflow models.

On the **IN/OUTPUT PARAMETERS** tab, variables are defined as input parameters, output parameters, or input and output parameters.

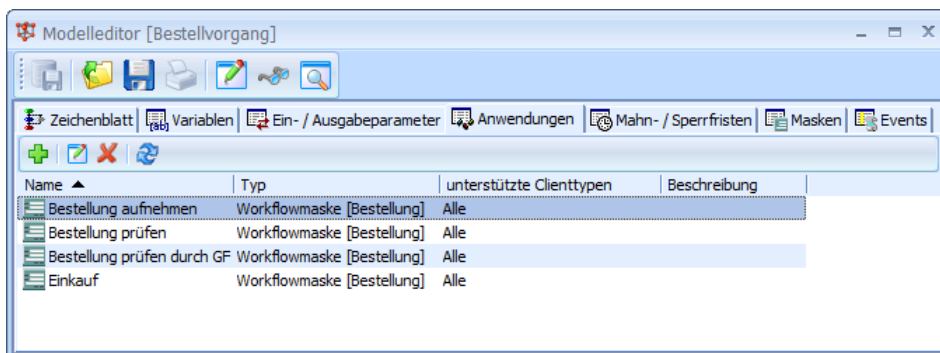


Applications

Applications are used to edit variables in individual activities. This can be achieved with automatically executed software instances, but usually with applications and user interaction. In enaio® the user interface and therefore many of the workflow applications are integrated in enaio® client. By default applications are associated with workflow forms in which process data can be changed.

As previously illustrated, it is necessary to specify application parameters used for data exchange between the workflow engine and the application.

Applications are set up on the **APPLICATIONS** tab.

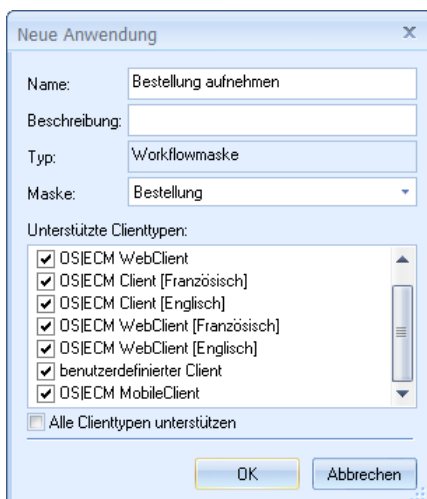


Set up an application for all activities executed by a user. Provided that different activities are executed with the same form and the same application parameters, an application can be assigned to multiple activities.

Currently, only applications which refer to a workflow form can be set up in the workflow editor.

Once a name has been entered for the application, select a form, previously designed with the form designer. It is useful to name forms, applications, and activities right away if they are destined to be used in the same activity.

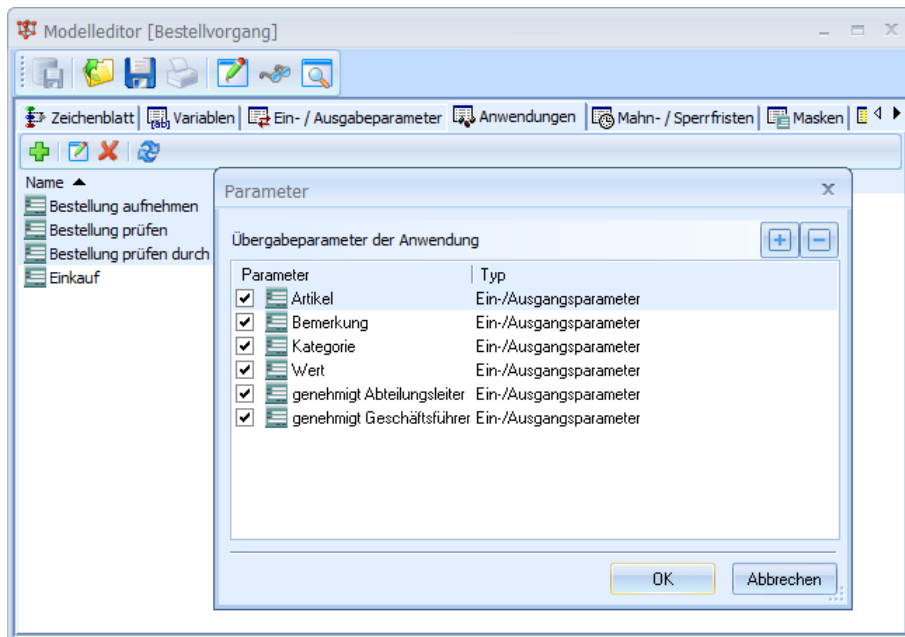
Depending on the client type, different applications can be defined. It is also possible to specify the same application for all available client types. While editing the application, specify to which client types the application will be assigned.



You can create applications for different languages that have the same function but forms with different labels, as well as applications for enaio® client or enaio® webclient to meet the requirements of different environments.

Setting application parameters

Once applications have been set up, create their parameters. The configuration dialog can be opened from context menu.



When depicting applications on forms, which is the standard case, the application parameters correspond to the form fields. Ensure that there is an application parameter for each form field. Field names are displayed in the dialog.

Choose a name and type for the new parameter. It can have one of the following values:

- Output parameter** the application (i.e. the form) transfers the value to the workflow system. The parameter value is not initialized. This is useful for input fields that have not been initialized before.
- Input parameter** the value is transferred to the application/form, but not returned to the workflow engine. This is useful for the display of static information or for fields that are no longer supposed to be modified by the activity.

Notice: input parameters are not marked on the form nor are they write-protected.

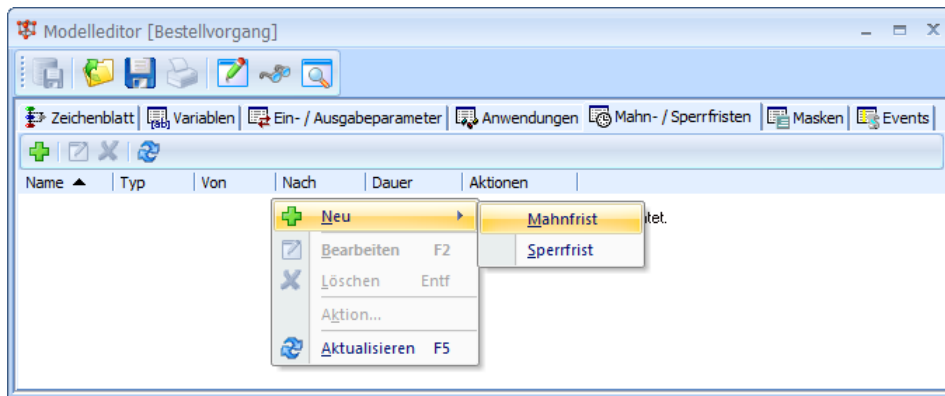
- Input/output parameter** the value is transferred to the form where it can be changed and then returned to the engine. This is useful for data to be changed during an activity.

All parameters must be defined for each application. Usually, you can stick to the workflow variable definition and the input form. Please note that a parameter must be created for each field, however not every variable must be transferred to the forms and neither every parameter must be edited with forms.

Dunning and Retention Periods

Dunning periods ensure that activities or sequences of activities are processed within a specific period of time. For a deadline, you have to specify which activity is executed if the period is exceeded.

Retention periods are used to specify that activities or sequences of activities cannot be completed before the end of a certain period of time.



Dunning periods

A new dunning period is set up by choosing the **NEW DUNNING PERIOD** from the context menu.

Select either the beginning or the end of any activity for configuring its duration.

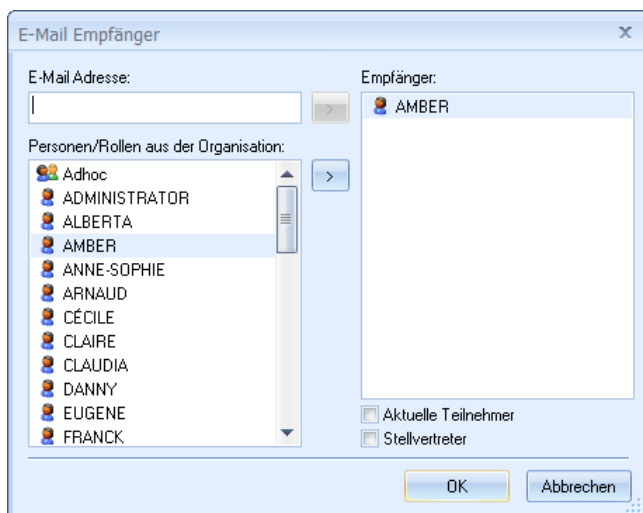
In enaio® client, an activity starts with its personalization and ends by forwarding after being processed completely.

Dunning periods can additionally be specified for multi-instance activities for the period between instance start and instance end. Instance start and instance end, however, are only available for the period within an instance.

The duration can be specified in hours, minutes and seconds.

An action is also defined for a dunning period. Available actions are sending e-mails and the assignment to users or roles. You can add an action using the context menu:

The **To** button on the **E-MAIL** tab opens a list with all roles and persons belonging to the organization. Persons/roles can be selected as recipients with arrow buttons or by double-clicking. Any other e-mail address can be entered, too. The **DEL** key removes selected recipients.



Use the e-mail button to set up further e-mail messages with a different subject line or text.

Already set up messages are removed with the delete button.

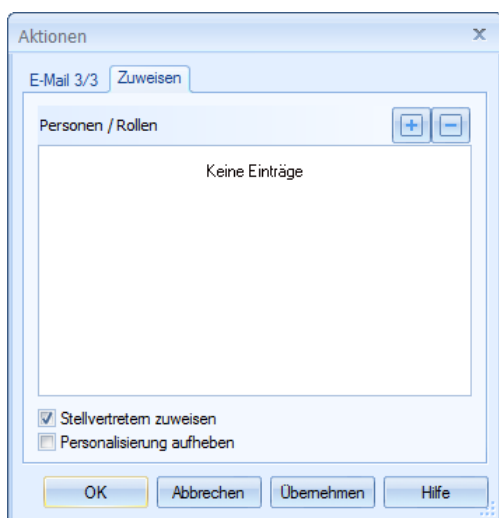


The following placeholders can be inserted into the subject line and the message field:

%OrganisationName	Organization name
%FamilyName	Workflow family name
%WorkflowName	Workflow model name
%ProcessName	Workflow process name
%DestinationActivityName	End activity that refers to the dunning period
%ReminderTime	Dunning period deadline
%BlockingActivityNames	a list of all current process activities, separated with commas and space characters, which were personalized but not yet forwarded or were placed in the inbox of the user, but not started yet.

Any character can be entered before or after placeholders.

Specify persons or roles that will have a current activity placed in their inboxes after the dunning period has been exceeded on the **ASSIGN** tab.



If the personalization is not removed, the activity is only moved to the inbox of the addressee. It remains flagged as personalized. The addressee can remove the personalization and start the activity himself. The user who has exceeded the dunning period can finish the activity as long as it has not been personalized by the addressee.

If the personalization is removed or if an activity is not personalized, the addressee as well as any other user can be entered as participants of this activity and execute it.

You can also assign this activity to substitutes.

Retention period

A new retention period is set up by choosing the **NEW RETENTION PERIOD** from the context menu.

Select the beginning of any activity for configuring its duration. The duration can be specified in hours, minutes and seconds.

Actions cannot be defined for retention periods. If a user wants to execute an activity before the retention period has expired, he will be notified in enaio® client and cannot execute nor complete the activity.

Drawing Page

Activities and transitions between activities are graphically defined on the drawing page.

Two activities have already been created by default:



The start activity is the starting point of the activity sequence. It has no properties to be configured.



The end activity is the end of the activity sequence. Joins and variables transfer are configured for the end activity.

You can create three types of activities:



User-controlled activities

Configure an application, participants, joins/splits, variable transfer, and the inbox for user-controlled activities.



Loop activities

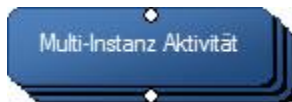
Configure the type 'Repeat/Until' or 'While', the corresponding condition, joins/splits, and variable transfer for loop activities.

Loops have no participants and no application.



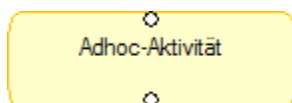
Route activities

Routes are supportive activities that merge procedures at joins/splits and loops. Configure joins/splits and variable transfer for routes.



Multi-instance activity

User-controlled activities are executed by exactly one participant. Multi-instance activities are executed by all assigned participants at the same time. The process does not continue until all participants have forwarded the process step.



Ad hoc activity

Ad hoc activities are areas that activities without predefined transitions are integrated in. Users deploy circulation slips in enaio® client to define the transitions between these activities. Circulation slips may define further processes and be designed in a way that the current user can edit further process flow.

Activities are linked by transitions. Transitions are depicted as arrows between the exit of an activity and the entry of another activity.

Create transitions from one activity to other activities and define a condition for each transition under which the transition is taken.

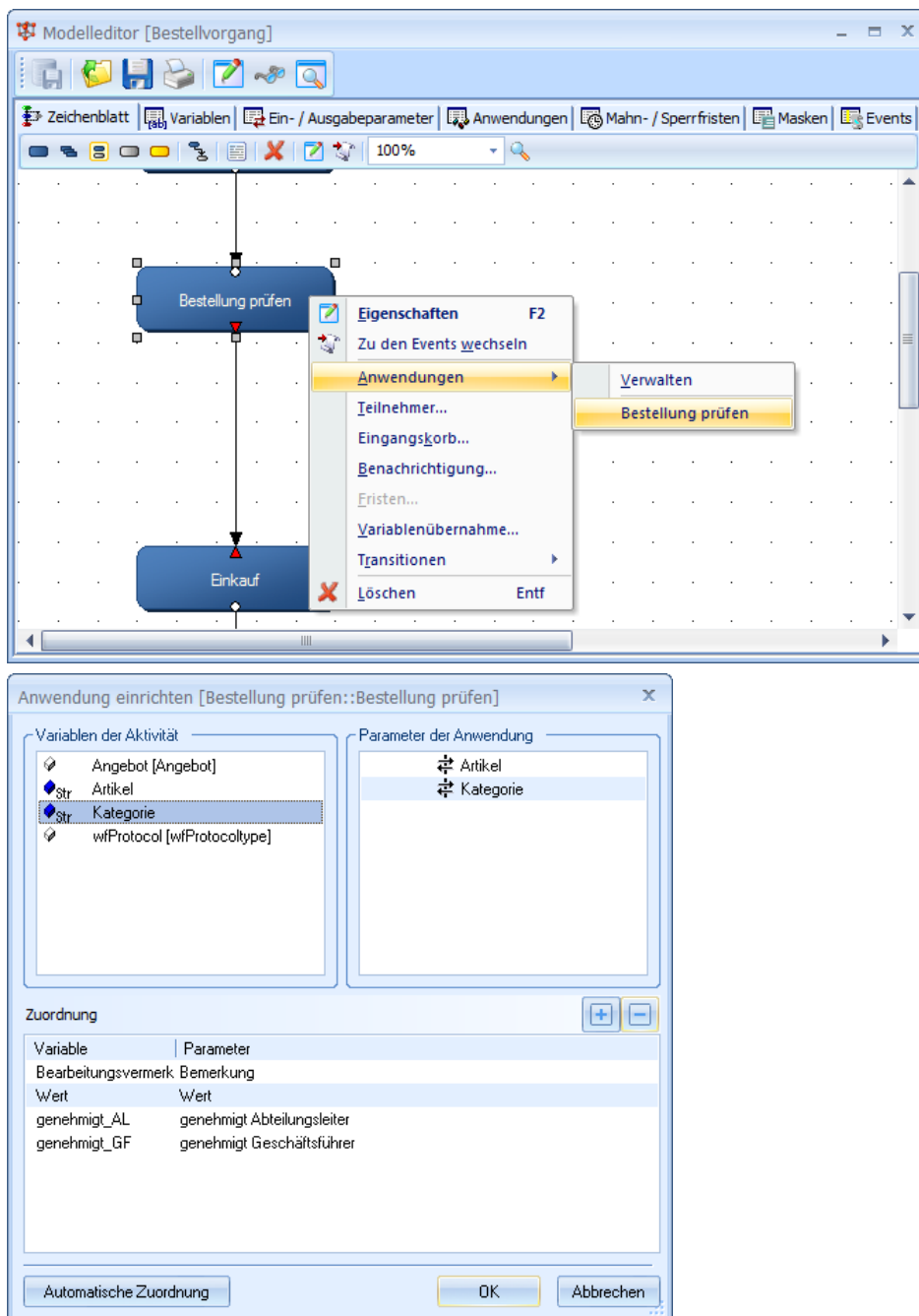
Loops have a loop exit and a loop entry. A transition of the loop exit to the first activity within the loop has the property 'From Loop'; a transition of the last activity within a loop to the loop has the property 'To Loop'. These transition properties are allocated automatically. Activities within an ad hoc activity do not have predefined transitions.

Activities and applications

As mentioned above, start applications to change process variables if variables are not manipulated within the workflow engine. For that reason you have to indicate an application with which each user-controlled activity is edited. By default the application is linked by a workflow form. You must also determine which variable is assigned to which application parameter.





It is possible to assign multiple applications to an activity if the application supports different client types. Applications are assigned to an activity using the context menu item **MANAGE APPLICATIONS**.

For each assigned application, application parameters of one variable are assigned at a time. There are different kinds of application parameters. Input parameters transfer data from the workflow engine, but do not return them. In contrast, output parameters are transferred to the workflow engine, but are not transferred from there. Input and output parameters transfer data in both directions.



Choose the intended application and assign variables to the application parameters.

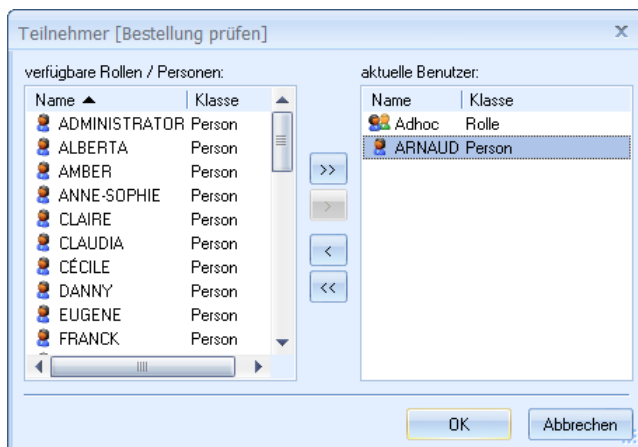
The parameters of the application are marked as follows:

-  Form parameters
- ab|** Parameters without form field
-  Input/output parameters
-  Output parameters
-  Input parameters

A variable must be assigned to each defined parameter. If this has not been specified in the model, the process is canceled at this activity with an error.

Activities and participants

Assign participants to each user-controlled activity using the context menu.



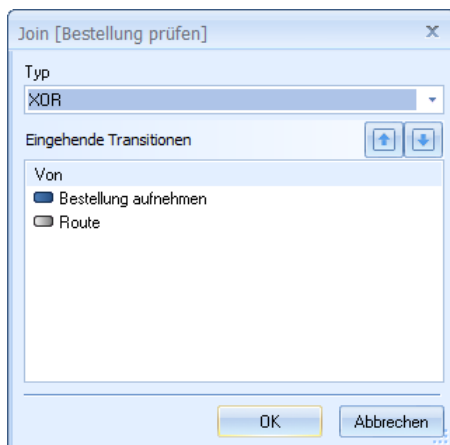
Roles and users can be chosen as participants. Roles and users are set up in the organization explorer.

Additionally, virtual roles are available. Virtual roles are roles, that are assigned to persons due to their function in a specific process operation. For example, the virtual role 'Process participants' contains all persons that already participated in the process.

Activities and joins/splits

In more complex process models, it is necessary to handle branching and merging of editing threads separately. This is realized with joins and splits.

Joins and splits can be defined using the context menu of an activity. In this example it would be possible to create a XOR join for the activity 'Purchase', i.e. the activity is only executed when exactly one of the incoming transitions has been completed. Due to an OR split this is already given, but with that example joins can be exemplified again.



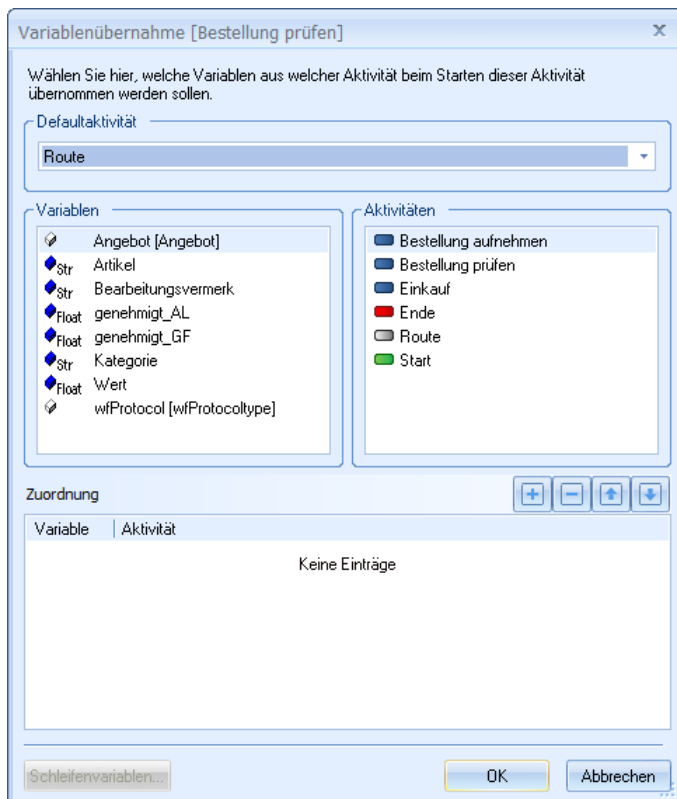
For the activity 'Purchase' all incoming transitions are listed in the dialog. Transitions of the type 'AND' are predefined. A list enables you to simply change the type.

Entry and exit points of activities in transitions of the type 'XOR' are marked red on the drawing page.

Activities and variable transfer

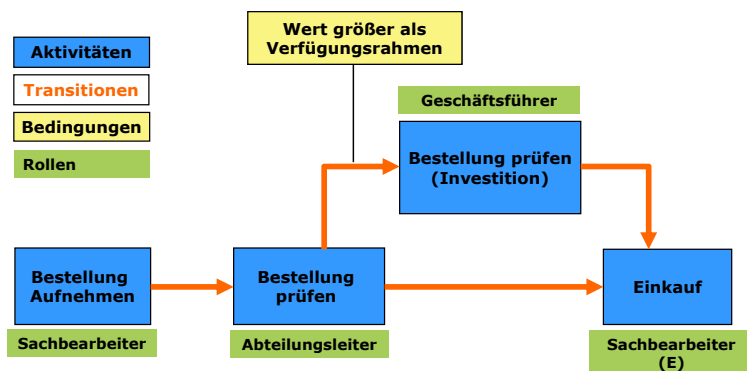
Variables can be given values in the workflow. Within an activity they are perceived as local variables. This difference is only relevant if different processing steps are executed simultaneously. In this case each editing thread has a local copy of the variables, which can be changed in each thread independently.

It is usually acceptable to exactly indicate one activity from which all variable values of this activity are inherited. The preceding activity is chosen in the dialog 'Variable transfer'. Except the start activity, a standard activity for variable transfer is assigned to each activity.



If the process involves multiple editing threads to be merged, for example after merging parallel process steps, you have to define from which activity variable values will be applied. Thereby it does not matter whether the editing threads have been executed parallel or exclusively. Important is that correct values of respective preceding activities are transferred to the variables.

This is shown in the example of the purchase order workflow (see chapter basics). There are two ways to access the activity 'Purchase'. On the one hand, the approval can be given directly by the department manager, though if it is about a more significant investment good, the approval of the chief executive officer has to be obtained.



In this case alternatively two ways in the process graph can be chosen. For the activity purchase, however all variables of the process have to be allocated with a valid value. The activity 'Purchase order' by default inherits these values, except the value of the variable 'approved (chief executive officer)'. As the department manager could not edit this field, this variable could not have been changed through the activity 'check purchase order'. For that reason this model has to provide that the variable will be inherited from the activity 'approved (chief executive officer)' by exception.

The specific assignment of the variable passing occurs selecting the intended variable and the respective activity followed by pressing the **ADD** button. Press the **REMOVE** button to remove an assignment.

According to the process model, variables may be transferred from more than one available activity. If so, multiple assignments of the variable transfer can be defined. The workflow engine processes the variable transfer list and, in case the respective activity has been performed, assigns values to the variable ignoring the rest of entries.

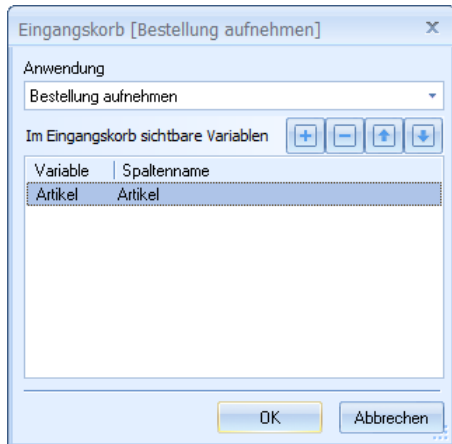
If no special assignment exists, values from standard activity are taken over. The standard activity must be performed; otherwise variables are provided with error entries instead of values. In this case, the result of further process flow is not defined.

Set up an extended variable transfer for loops (see page 39).

Activities and the inbox

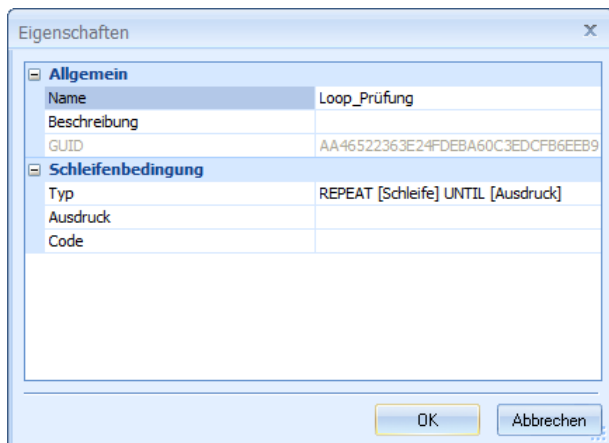
In their inboxes participants find all activities they can execute. Furthermore, the inbox can display the current value of variables for each activity. That allows the user to better distinguish between the running activities of a model.

Configure the display of variables in the inbox using an activity's context menu.



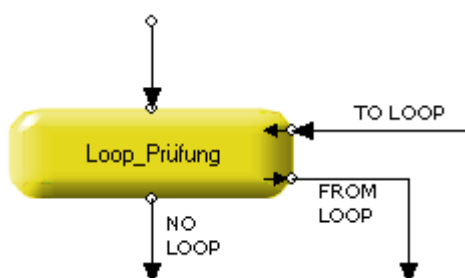
Type, condition and transition of loops

Define the loop type and a condition over the properties dialog in the context menu for loops.



Transitions of the loop type 'Repeat/Until' to the following loop activity are taken until the condition is fulfilled. The transition to the following loop activity has the property 'From loop'. If the condition is fulfilled, the transition with the property 'No Loop' will be taken.

Transitions of the loop type 'While' to the following loop activity are taken as long as the condition is fulfilled.



Enter the condition into the **Expression** field.

Allgemein	
Name	Loop_Prüfung
Beschreibung	
GUID	AA46522363E24FDEBA60C3EDCFB6EEB9
Schleifenbedingung	
Typ	REPEAT [Schleife] UNTIL [Ausdruck]
Ausdruck	genehmigt_Geschaeftsfuehrer.value = 1
Code	

Comments cannot be entered in the **Expression** area as they will lead to errors.

In the **Code** row, you can enter script code that is executed before the expression has been validated. Therewith, you have the possibility to recalculate the values of the variables or specify decision criteria by further checks.

Detailed loop examples can be found in the section 'Loops'.

Transition conditions

Create transitions from one activity to several other activities and define a condition when to take each transition. The taken transition has to fulfill the conditions completely.

A transition condition must be entered using the properties dialog of a transition in the **Expression** row.

Allgemein	
GUID	CC4853471D5D485B8ABE1628
Von	Bestellung aufnehmen
Nach	Route
Typ	NO LOOP
Ausdruck	Wert.value >= 400
Code	

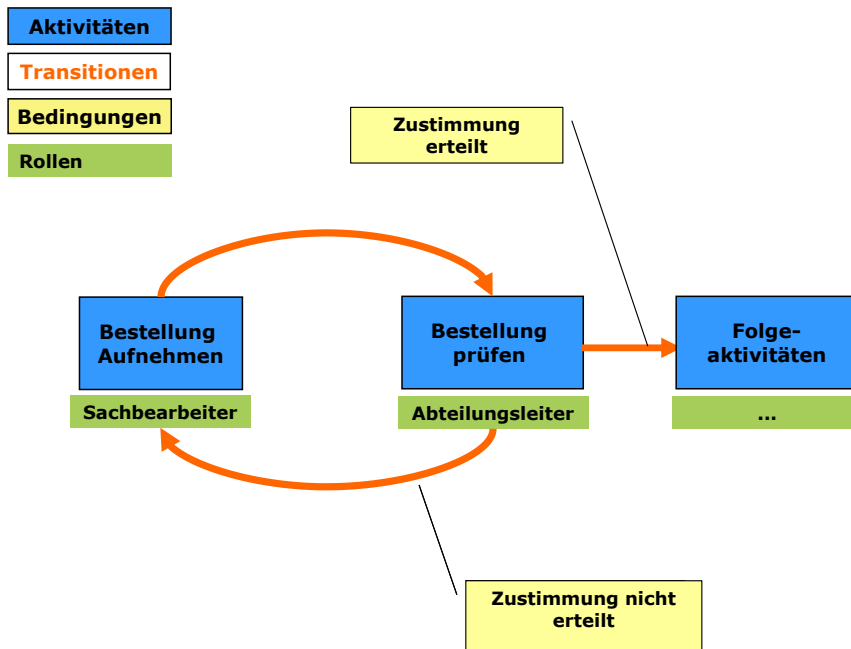
In the **Code** row, you can enter script code that is executed before the expression has been validated.

Extended Process Modeling

Loops

As mentioned above, it may become necessary to repeat individual activities or whole parts of a process model until a specific condition is fulfilled. This is depicted in the process model with loops which stand for special activities.

The current example may include that the department manager wants to query more information from the ordering person before giving his approval and such follow-up inquiries must be depicted with the workflow model. Since the number of queries is unknown in the first place, a loop must be designed. The decision whether a process is forwarded to the purchase department or the chief executive officer is supposed to depend on the approval of the department manager. If this is not the case, the process is returned to the ordering person as illustrated in the image.



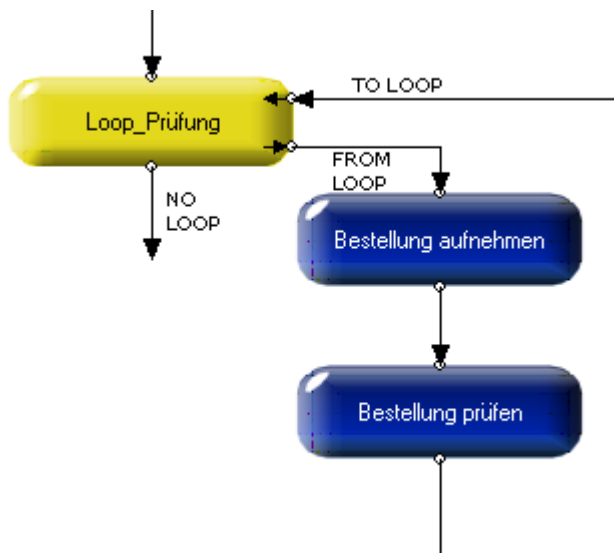
The possibilities offered by a higher programming language in which the process flow can be verbalized using statements like the following are not available during process design.

```
while <Condition>
{
  // Statement
}
```

```
repeat
{
  // Statements
}
```

Therefore a workaround, special loop activities, are used here. A special characteristic of loops is that they appear as a unit for activities outside the loop and that they can contain any number of activities inside the loop. The loop activity is thus a kind of container for the activities inside the loop. Each loop activity has a condition that decides whether or not to process or repeat the loop.

The following again visualizes the mechanism of loop activities. A start activity (not displayed) creates a transition for the entry of the loop activity (Loop check). From the loop activity the first activity "take purchase order" is addressed over the loop exit and from there the activity "check purchase order". Depending on loop type the condition is either evaluated before loop entry or after loop performance, which then decides whether or not to repeat the loop.



Loop activities are either of the type **while** or **repeat/until**. The type determines if the condition is checked before entering the loop or if the condition will not be evaluated until the loop is performed at least once.

The conditions of 'While' loops is checked before the loop branches. 'While' loops can be associated with the following statement: repeat the loop as long as the condition is true.

'Repeat/Until' loops are performed at least once. Afterwards it is checked whether the loop has to be repeated. This is the case if the specified condition has not been fulfilled yet. You can remember the mechanism of a 'Repeat/Until' loop with the following explication: repeat the loop at least one time, but until the condition is true.

In our example the loop condition would be that the department manager has not confirmed yet. As the variable is an integer that accepts the values 0 (FALSE) or 1 (TRUE), conditions have to be verbalized accordingly. Formulate the conditions as follows for while-loops:

```
approved_DepartmentManager.value = 0
```

The loop is performed as long as the variable value approved_DepartmentManager equals 0, thus has not been approved by him yet. The value must be preset with 0 for the while-loop in our example, of course, as after taking the purchase order no decision has been made yet.

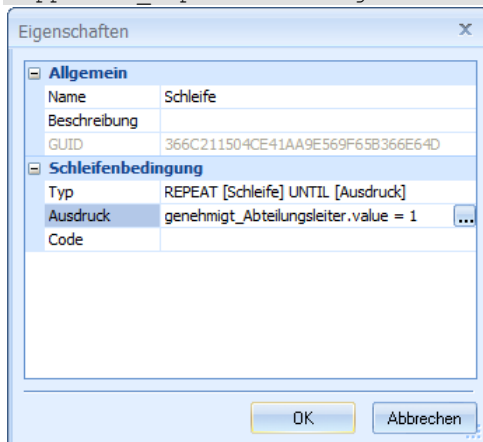
Though, deploy a repeat/until-loop here as the inside of the loop would have been processed at least once. The related condition is here:

```
approved_DepartmentManager.value = 1
```

The loop thus has to be processed until the department manager has given his confirmation.

Determine the condition in the **Expression** row on the properties dialog of the loop activity. Since a repeat/until-loop has been chosen here, enter the following condition:

```
approved_DepartmentManager.value = 1
```



Add further transitions connecting the loop activity with external and internal activities to ensure that the loop is performed. The transition type is specified according to the transition. Transitions that connect loop activities with internal activities are separately flagged. The types 'From Loop' and 'To Loop' are available for this purpose. In this special case these are transitions connecting the loop activity 'Loop_Check' with the activity 'Take purchase order' (From Loop) and the activity 'Review purchase order' with the loop activity (To Loop). This is visualized above.

In summary the following transitions have to be set up:

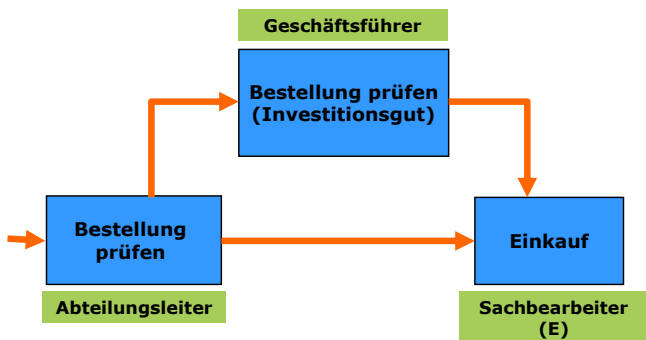
From activity	To activity	Type
StartActivity	Loop_Check	no loop
Loop_Check	Take purchase order	from loop
Take purchase order	Review purchase order	no loop
Review purchase order	Loop_Check	to loop
Loop_Check	Review purchase order (investment goods)	no loop
Loop_Check	Purchase	no loop
Review purchase order (investment goods)	Purchase	no loop
Purchase	EndActivity	no loop

The transition type is automatically assigned. Entry and exit points of the inner activities are marked with arrows. 'From Loop' transitions are attached to the exit arrow of a loop, 'To Loop' transitions lead to the entry arrow of a loop.

Variable Transfer in Loops

Variable transfer for activities is required to decide which values must be transferred for all subsequent activities. The first activity that does not have any activity in front can be defined as standard activity from which the values can be copied. As soon as there are multiple transitions leading to an activity, the engine has to decide from which preceding activity variables must be copied. This process cannot be automated as variables can take over different values on different threads. For that reason, you have to specify the activity order during process design for possible variable transfer.

The 'Purchase' activity example explains this content very well. There are two ways leading to the activity 'Purchase'. Variables can be taken over from all preceding activities, thus not necessarily from the direct predecessor. But usually values that might have changed in the preceding activity are intended to be copied to the following activity. Define the order of preceding activities so variables are transferred correctly for each transition.



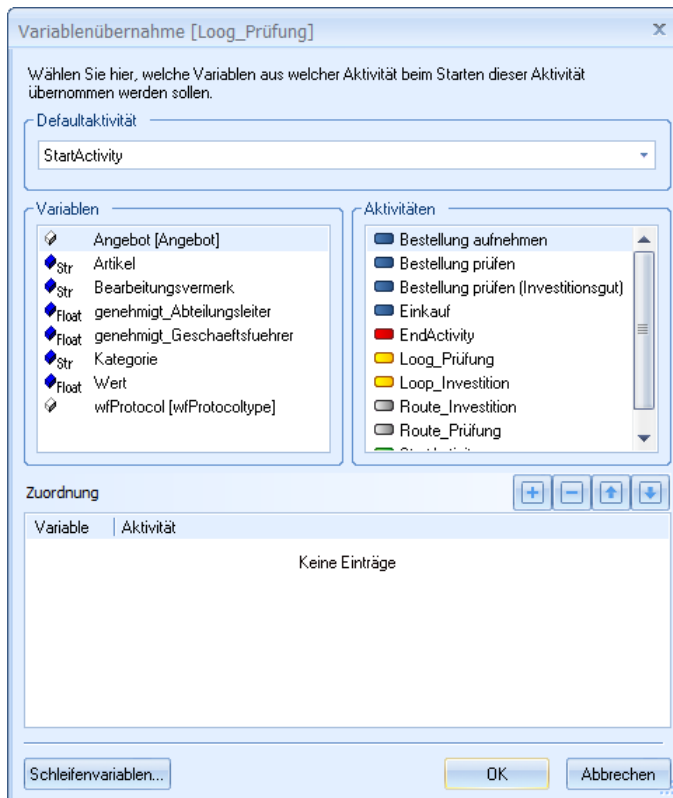
Here 'Review purchase order' and 'Review purchase order (investment)' are preceding activities of the 'Purchase' activity. The 'Review purchase order' activity is set as standard activity and an explicit variable transfer from the 'Review purchase order (investment good)' activity is defined. In this specific case, the order is of no importance and the 'Review purchase order (investment good)' activity could have been set as default as well.

Variable transfer for loops is much more complex than for usual activities. The process designer has to decide whether to re-set the variables for each performance as at the beginning of the loop or to let the variables change within the loop keeping them during the next performance.

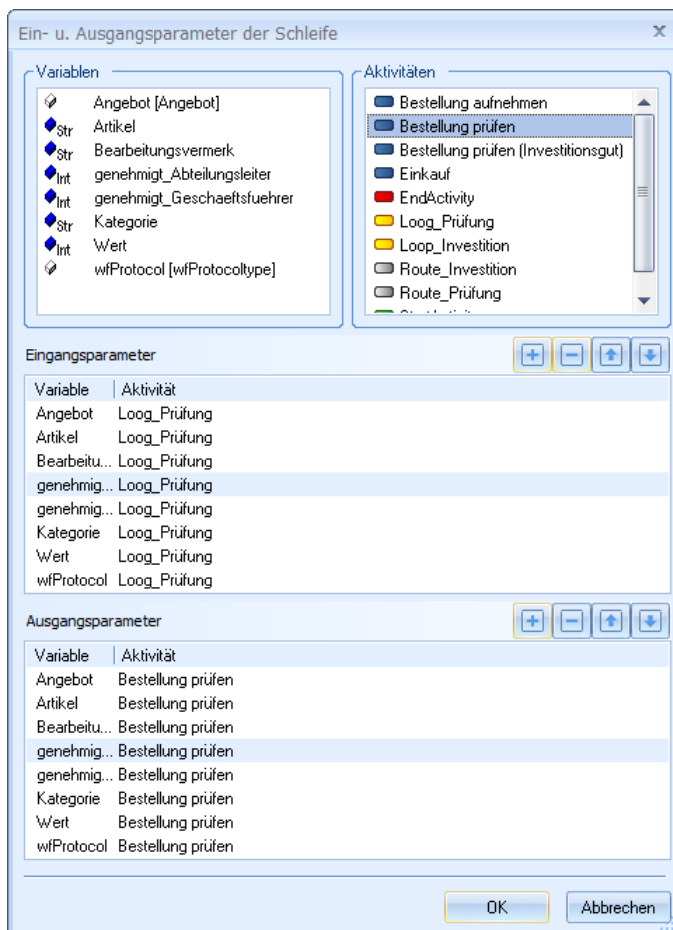
Specific assignments to variables are evaluated according to the indicated order. Indicated variable values of the preceding activity that corresponds to the assignment are taken over. Otherwise, the variable values of the standard activity are transferred.

In our example the variable transfer of the loop activity 'Loop_Check' is viewed.

In the first step, determine the variable transfer of the loop activity 'Loop_Check' for the moment the loop is reached for the first time. In the example, this occurs over the transition 'StartActivity-Loop_Check'. Thus, the 'StartActivity' activity is set as standard activity. Further assignment is not required as data are transferred directly from the preceding activity when reaching the loop for the first time.



All further loop variables are set as input or output parameters. Click on the **LOOP VARIABLES** button to open the dialog.



Input parameters are variables that are changed by the activities within the loop and that are available in the modified form when again entering the loop. The loop itself is the last activity that edits the variables before they enter the loop again and verifies whether or not the exit condition is fulfilled. The example input parameters, thus are variables that come from the activity 'Loop_Check' and are forwarded to the activity 'Take purchase order' in modified form.

Output parameters are the variables that are forwarded to the next activity when exiting the loop. Variables were allocated in the last activity within the loop. In this example, the output parameters come from the activity 'Review purchase order'.

More Complex Loops

In the above presented example, a loop returns the process from the activity 'Review purchase order' to the activity 'Take purchase order' until the department manager has no further questions and confirms the process. In practice, this workflow might include a loop returning the process from the activity 'Review purchase order (investment)' to the activity 'Review purchase order' thereby allowing the chief executive officer to send further inquiries to the department manager.

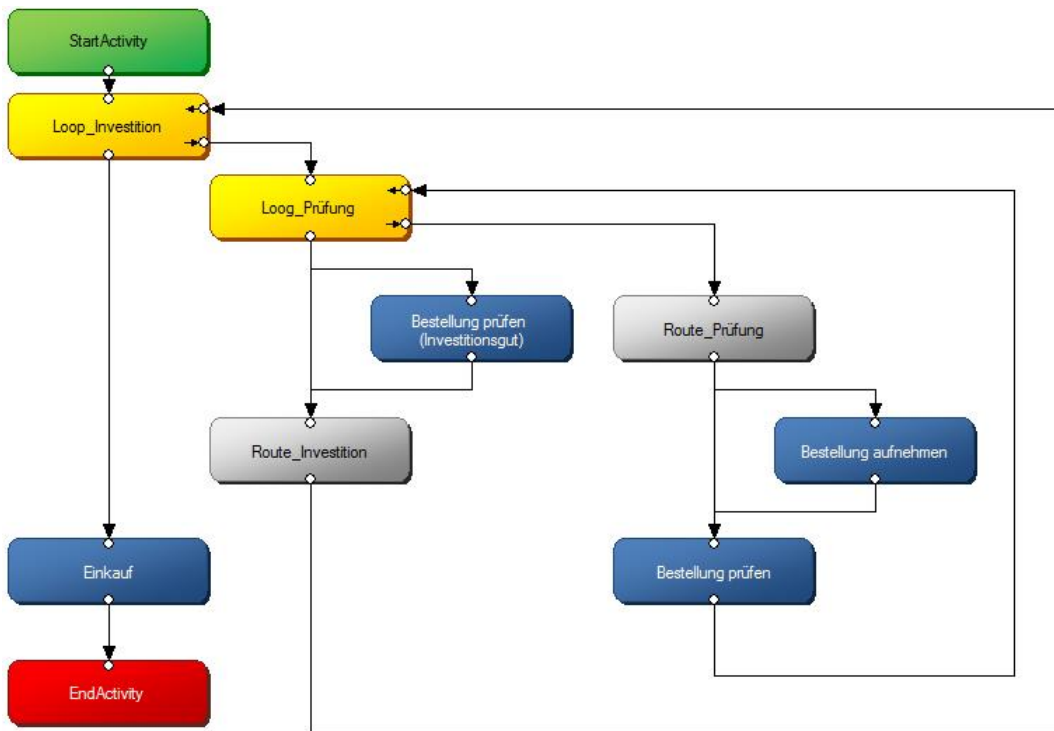
There are two possible solutions for this requirement. For the one thing, another loop can be deployed and for the other thing, further branches can be used within the loop.

For both solutions, activities of the type 'Route' are used. These support activities join threads and are necessary as no conditions are available for 'To Loop' and 'From Loop' transitions. The first and the last activity within a loop must always be performed.

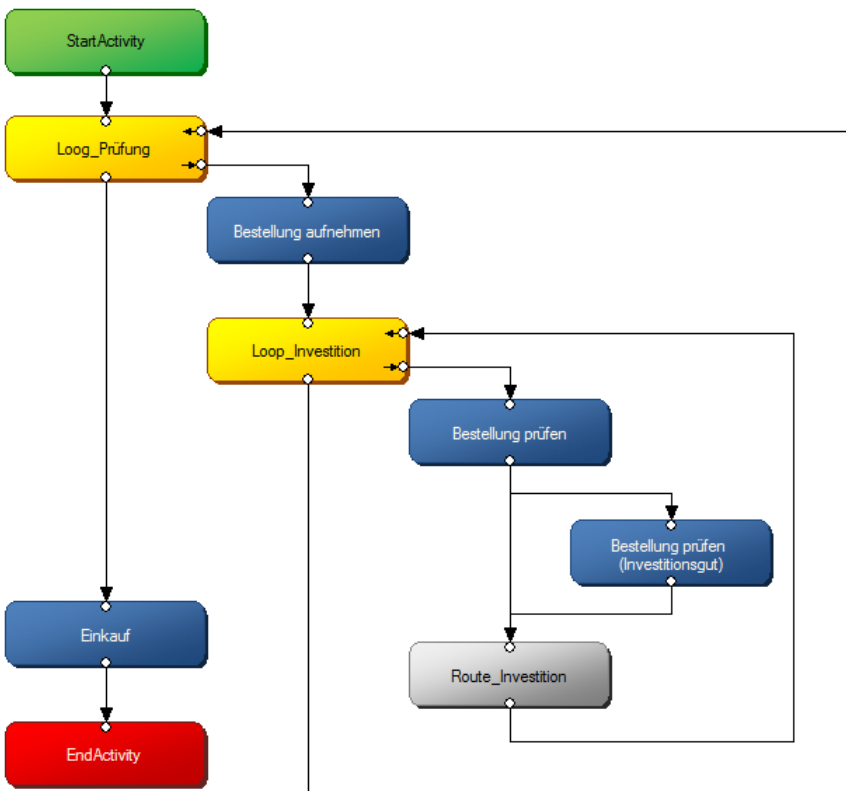
Nested Loops

One solution may be another integrated loop for further inquiry. There are two possibilities available.

Model A:



Model B:



Model B is less complex as it only requires one supporting 'Route' activity and one thread with transition conditions. In the following, this model is explained in greater detail.

Both models allow the integration of a backlink from the activity 'Review purchase order (investment)' to the activity 'Take purchase order' to enable the chief executive officer in the activity 'Review purchase order (investment)' to undo the confirmation given by the department manager in the activity 'Review purchase order'.

Activities

The following activities are required:

Activity	Type	Conditions
StartActivity	Procedure	
EndActivity	Procedure	
Take purchase order	Application	
Review purchase order	Application	
Review purchase order (investment)	Application	
Purchase	Application	
Loop_Investment	Loop (Repeat/Until)	Loop condition: approved_ChiefExecutiveOfficer=1 Or Value.value<400 Or (Value.value>=400 And approved_DepartmentManager.value=0)
Loop_Check	Loop (Repeat/Until)	Loop condition: approved_DepartmentManager.value =1
Route_Investment	Route	

The loop 'Loop_investment' enables further inquiry between the chief executive officer and the department manager. The loop is not left until one of the following conditions is fulfilled:

- approved_ChiefExecutiveOfficer=1
The chief executive officer has given his approval.
- Value.value<400
The value is less than € 400. The chief executive officer's authorization is not required.
- (Value.value>=400 And approved_DepartmentManager.value=0)
The value is greater than or equals € 400, but the department manager's authorization has not yet been obtained.

As all processes pass through the loop, processes that do not or not yet affect the chief executive officer also have to leave the 'Loop_Check' activity loop.

The loop 'Loop_Check' enables further inquiry between the department manager and a person in charge. The loop is not exited towards the activity 'Purchase' until the following condition is fulfilled:

- approved_DepartmentManager.value =1
The department manager has given his approval.

More loop conditions are not necessary as processes that have been approved by the department manager and that additionally require the chief executive officer's approval do not reach the loop until both approvals are present. The process remains in the loop 'Loop_Investment' until the chief executive officer has given his approval.

The route activity 'Route_Investment' is necessary as only one 'To Loop' transition leading to the loop 'Loop_Investment' is permitted and data for this loop may come from the two activities 'Review purchase order' and 'Review purchase order (investment)'.

Transitions

The following transitions are required:

from	to	Type
StartActivity	Loop_Check	no loop
Loop_Check	Take purchase order	from loop
Take purchase order	Loop_Investment	no loop
Loop_Investment	Review purchase order	from loop
Review purchase order	Review purchase order (investment)	no loop Transition condition: Value.value>=400 And approved_DepartmentManager.value=1
Review purchase order	Route_Investment	no loop Transition condition: Value.value<400 Or (Value.value>=400 And approved_DepartmentManager.value=0)
Review purchase order (investment)	Route_Investment	no loop
Route_Investment	Loop_Investment	to loop
Loop_Investment	Loop_Check	to loop
Loop_Check	Purchase	no loop
Purchase	EndActivity	no loop

Starting from the activity 'Review purchase order' there are two transitions:

- The transition towards the activity 'Review purchase order (investment)' is taken if the value is greater than or equals € 400 and the department manager has given his approval.
- The transition towards the activity 'Route_Investment' is taken if the value is less than € 400 or the value is greater than/equals € 400, but the department manager has not yet given his approval.

Transition conditions must be defined in such a way that one of the available transitions is taken.

Variable Transfer

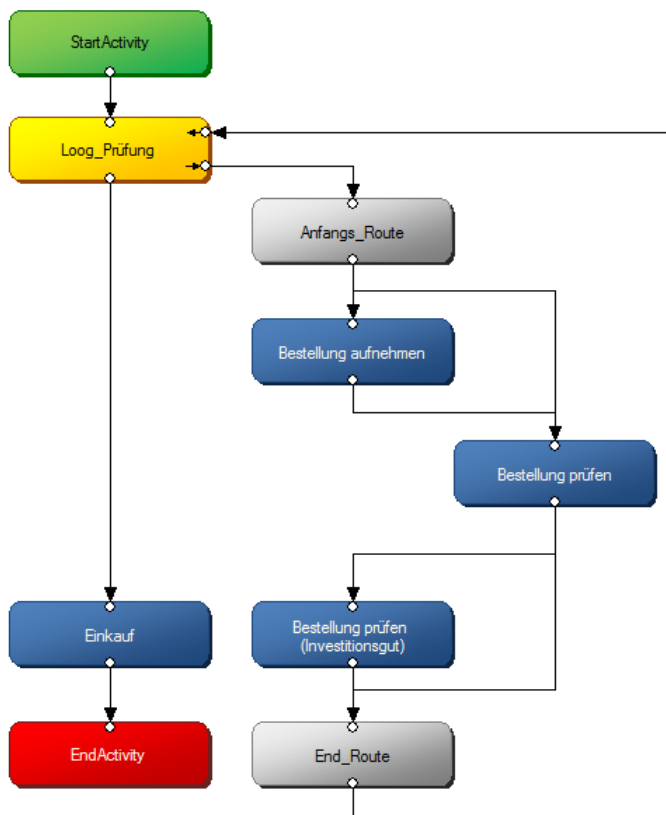
The following variable transfers are required:

Activity	Standard activity	Assignment
StartActivity	-	
EndActivity	Purchase	
Take purchase order	Loop_Check	
Review purchase order	Loop_Investment	
Review purchase order (investment)	Review purchase order	
Purchase	Loop_Check	
Loop_Investment	Take purchase order	Loop variables Input parameter: Loop_Investment Output parameter: Route_Investment
Loop_Check	StartActivity	Loop variables

		Input parameter: Loop_Check Output parameter: Loop_Investment
Route_Investment	Review purchase order	Review purchase order (investment)

Intersection within a Loop

Instead of deploying two nested loops, you can insert intersections with transition conditions into loops in order to fulfill the required inquiries between the department manager and a person in charge and between the chief executive officer and the department manager.



It is also possible to integrate a backlink from the activity 'Review purchase order (investment)' to the activity 'Take purchase order' to enable the chief executive officer in the activity 'Review purchase order (investment)' to undo the confirmation given by the department manager in the activity 'Review purchase order'.

Activities

The following activities are required:

Activity	Type	
StartActivity	Procedure	
EndActivity	Procedure	
Take purchase order	Application	
Review purchase order	Application	
Review purchase order (investment)	Application	
Purchase	Application	
Loop	Loop (Repeat/Until)	Loop condition: approved_ChiefExecutiveOfficer.value=1 Or (approved_DepartmentManager.value=1 And Value.value<400)
Start_Route	Route	
End_Route	Route	

The loop 'Loop' enables further inquiry between chief executive officer and the department manager as well as between the department manager and the person in charge. The loop is not left towards the activity 'Purchase' until one of the following conditions is fulfilled:

- approved_ChiefExecutiveOfficer=1
The chief executive officer has given his approval.
- (approved_DepartmentManager.value=1 And Value.value<400)
The department manager has given his approval and the value is less than € 400. The chief executive officer's authorization is not required in this case.

The route activity 'Start_Route' is necessary as the loop only allows one 'From Loop' activity, thus transition conditions cannot enable the activity to be forwarded to the activities 'Take purchase order' and 'Review purchase order'.

The route activity 'End_route' is necessary as only one 'To Loop' transition towards the loop is allowed and data for this loop may come from the activities 'Review purchase order' and 'Review purchase order (investment)'.

Transitions

The following transitions are required:

from	to	Type
StartActivity	Loop	no loop
Loop	Start_Route	from loop
Start_Route	Take purchase order	no loop Transition condition: approved_DepartmentManager.value =0
Start_Route	Review purchase order	no loop Transition condition: approved_DepartmentManager.value =1
Take purchase order	Review purchase order	no loop

Review purchase order	Review purchase order (investment)	no loop Transition condition: Value.value>=400 And approved_DepartmentManager.value=1
Review purchase order	End_Route	no loop Transition condition: Value.value<400 Or approved_DepartmentManager.value=0
Review purchase order (investment)	End_Route	no loop
End_Route	Loop	to loop
Loop	Purchase	no loop
Purchase	EndActivity	no loop

Starting from the activity 'Start_Route' there are two transitions:

- The transition towards the activity 'Take purchase order' is taken if the department manager has not yet given his approval.
- The transition towards the activity 'Review purchase order' is taken if the department manager has already given his approval.

Starting from the activity 'Review purchase order' there are also two transitions:

- The transition towards the activity 'Review purchase order (investment)' is taken if the value is greater than or equals € 400 and the department manager has given his authorization.
- The transition towards the activity 'End_Route' is taken if the value is less than € 400 or the department manager has not yet given his approval.

Variable Transfer

The following variable transfers are required:

Activity	Standard activity	Assignment
StartActivity	-	
EndActivity	Purchase	
Take purchase order	Start_Route	
Review purchase order	Start_Route	Take purchase order
Review purchase order (investment)	Review purchase order	
Purchase	Loop	
Loop	Take purchase order	Loop variables Input parameter: Loop Output parameter: End_Route
Start_Route	Loop	
End_Route	Review purchase order	Review purchase order (investment)

Multi-Instance Activities

Within poll or acknowledgement processes it may be useful to present an activity to more than one participant at the same time. In a process flow, multi-instance activities are made available in the inboxes of all participants and can be

processed simultaneously. The transition towards the next activity is not taken until the all participants have forwarded the process step.

Multi-instance activities usually require data of single instances to be merged by an event. Local variables are used within an instance, a script determines and evaluates the local variable values of all instances and forwards them to the following activity.

Example:

All participants of an activity positively or negatively evaluate the activity using radio buttons.

An EndActivity event runs this evaluation. If there are more positive than negative votes, the variable 'result' receives the value '1'

```
lCount = thisprocess.CurrentActivity.MultiInstances.Count
lpos = 0
lneg = 0
for i = 0 to lCount -1
if Clng ( thisprocess.CurrentActivity.MultiInstances.Item(i) .
    GetDataFieldByName("ranking").Value) = 0 then
    lpos = lpos +1
else
    lneg = lneg +1
end if
next
if lpos > lneg then
    result.Value = 0
else
    result.Value = 1
end if
```

This result can be displayed in the following activity with the variable 'result'.

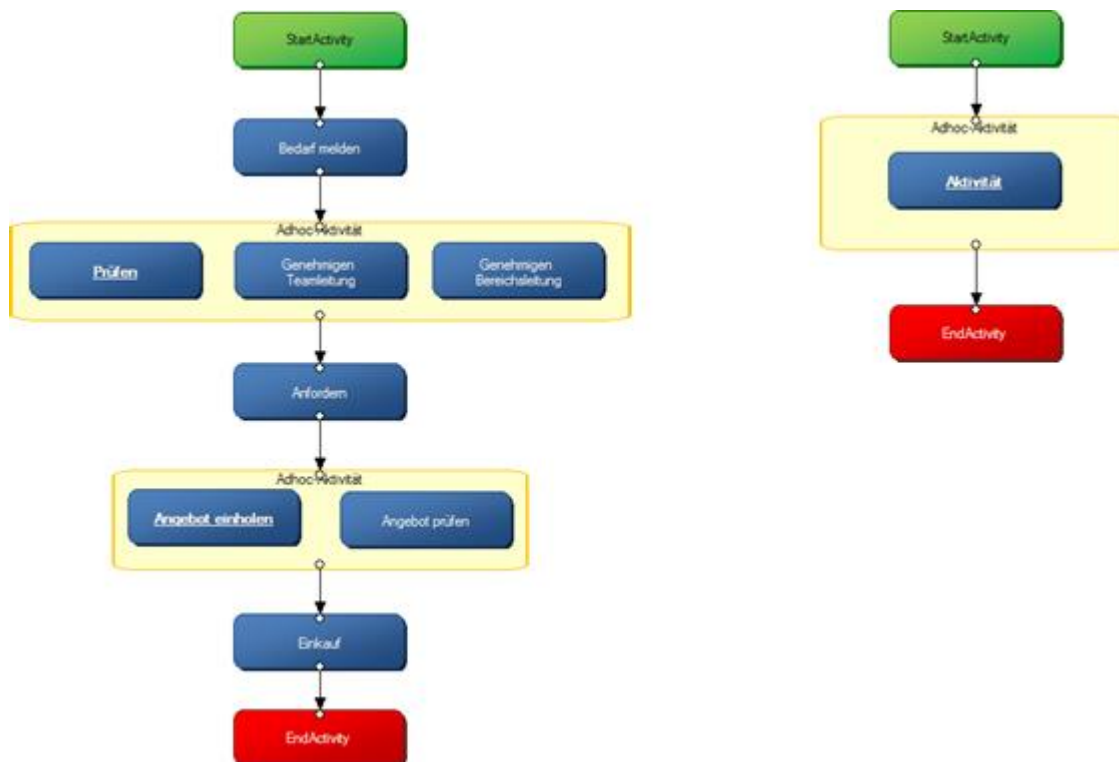
Further instances can now be added to multi-instance activities which are already running.

Ad Hoc Activities

Operating processes might be that complex that their depiction in a workflow model, the models' maintenance and customization would be very time-consuming and complicated. Ad hoc activities allow you to define model areas in which users can adjust processes to current situations. This simplifies the development and maintenance of models.

Any number of ad hoc activities can be added to a model at any position. Alternatively, a model can consist of a single ad hoc activity only. Within an ad hoc activity, at least one activity must be added but loop activities or further ad hoc activities are not required.

Examples:



One activity within the ad hoc activity is defined as standard activity. It will start first and is the one from which the user starts to configure the order of the following activities in a circulation slip. The user can thereby select all activities from the ad hoc area multiple times and also allocate them more than once in the circulation slip. In addition, the user decides whether or not the following users can change the entire circulation slip or single parts of it during the process flow.

The activity order is not defined in the workflow area so global variables will forward the data; otherwise they have to be organized with events.

Events

enaio® client and enaio® webclient differ in how they use client scripts. Information about events in enaio® webclient can also be found in the 'Workflow Event Scripts for enaio® webclient' handbook. The document is located in the directory 'Win32\Disk1\OSWEB\Doc' on the installation DVD. The following information in the specific examples and references relates to enaio® client.

It is not possible to call workflow jobs from server-side workflow events through the 'running context'. This causes the enaio® server to crash.

It would often be preferable if some process steps are executed automatically when processing activities. This can partly be realized in the model, but some actions must sometimes be executed out of a specific context. That is why both the client and the workflow engine allow you to start script code at specific events that perform these routines automatically. The interface is similar to the DMS events. Frequent application scenarios in workflow context are:

- Initializing data fields in forms
- Validity check of linked fields
- Integrity check during data entry
- Automatic editing of the workflow file when creating, adding or deleting documents
- Transferring DMS objects (folders, registers, documents) to or from the DMS
- Checking authorizations for value setting in the form
- Checks before an activity starts in the workflow engine
- Evaluation of organizational structure
- Script-controlled determination of activity editors
- Script-controlled creation, editing, and deletion of deadlines

There are multiple entry points for event code execution. The code can be written in VB Script. Therewith, extensive application possibilities are offered. The following objects can be addressed out of the script code by default:

- All objects of the Windows Script Host (File System Object, System Object)
- Database objects with Active Data Objects (ADO)

In client scripts the following COM interfaces are additionally available:

- The COM interface of enaio® client (see COM interface documentation)
- The oxactive.dll interface to access the transferred files (see documentation Event Editor)
- COM interface extensions to access the running process and the current activity
- Methods of the function library oxvbbas.dll registered in the client directory

The workflow engine provides server scripts with an interface to access the organizational structure, the running process and its variables, the deadlines, and the workflow file. Thereby, some object variables are already initialized and transferred to the script.

All available objects are listed in the workflow programming reference that can be found in the appendix.

Scriptable COM interfaces allow you to integrate further objects. Client scripts do often address Office applications in order to create documents in or query data from them. To run separate actions that cannot be displayed with script code, create ActiveX DLL files with a more sophisticated programming language (e.g. VB, C++, or Delphi) and integrate them with these scripts. The interface must support scripts. You must take that into account when designing the interfaces and transferring strings.

Objects that will be integrated in the server code are subject to separate guidelines and must be created by OPTIMAL SYSTEMS only.

The following events are available for activities:

Event	Execution	Entry point	Example of use
ButtonClick	Client	When clicking a button in a workflow form	Creation of a document with script code
StartActivity	Server	After creating an activity but before editing with an application	Variable initialization at the server
BeforeOpen	Client	Before the activity form is being depicted	Variable initialization, presetting fields with values, automatic document transfer from DMS/to DMS
EndActivity	Server	After editing an activity with an application but before reaching the actual condition	Dynamic variable modification because of workflow variables and data available at the server
BeforeForward	Client	After pressing the FORWARD button but before canceling the activity. The action can be canceled with return code. See below.	Plausibility check, document filing into the DMS
Before Cancel	Client	After the button CLOSE has been clicked.	
SimulateMaskEdit	Stimulation tool	Is used by a simulation tool for the application editing simulation.	Is not needed for productive systems.
PersonalizeWorkItem	Server	After a user has restarted a non-personalized process step and before this process step is personalized.	
GetWorkItemParams	Server	After a user has restarted a personalized process step and the variables have been read in.	
StartInstance	Server	After a user has started a multi-instance activity.	Variable administration for multi-instance activities

Event	Execution	Entry point	Example of use
EndInstance	Server	After a user has forwarded a multi-instance activity.	Variable administration for multi-instance activities
TimerFired	Server	After the dunning period of the corresponding activity has expired.	

If implemented, the events are executed in the following order.

Event	Place
StartActivity	Server
PersonalizeWorkItem	Server
GetWorkItemParams	Server
BeforeOpen	Client
ButtonClick	Client
BeforeForward	Client
EndActivity	Server

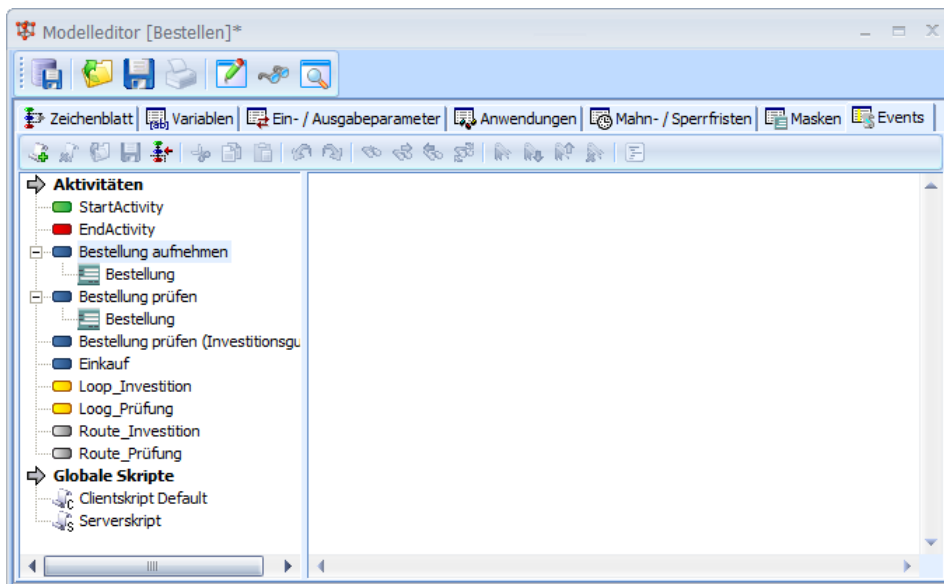
Special Features of the BeforeForward Event

With the BeforeForward event the editing of an activity can be controlled. According to the result of the validity check, it is afterwards determined whether to forward the activity or to cancel the process. Further procedure can be determined with the return code. The following values are available:

Returncode	Meaning
0	Activity can be forwarded
-1	Cancel forwarding, form stays open, data will not be saved
-2	Cancel forwarding, form will close, data will be saved
-3	Cancel forwarding, form will close, data will not be saved

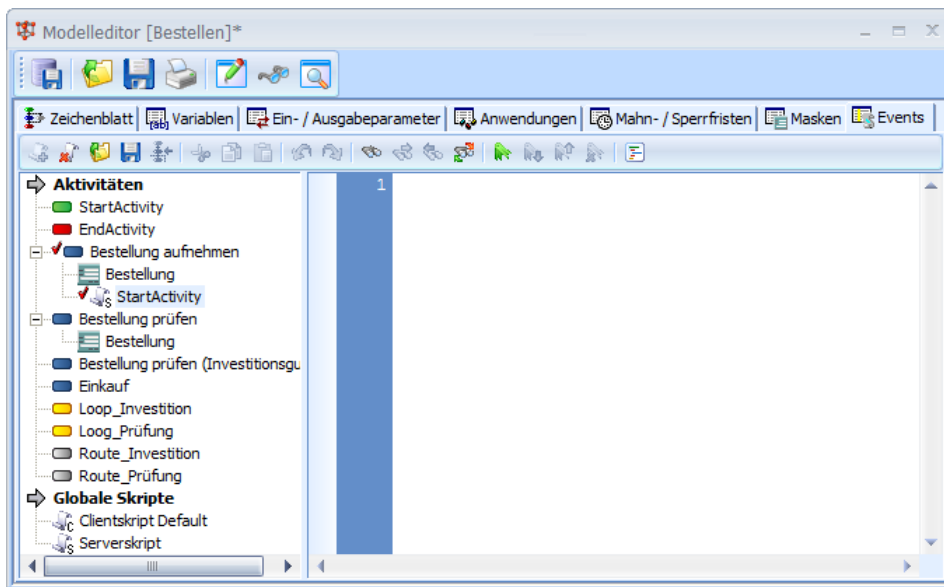
Setting up Event Code

Use the **EVENTS** tab in the model editor to set up events.



The event window shows all contained activities and their applications. Use the activity's context menu to add server events, and the application's context menu to add client events can. According to the application (and therewith

associated client types), there may be distinct client events. Event code can be edited in the right editing window which is activated after selecting an event.



Intended script code can be entered in the editing window. Thereby, syntax highlighting and auto complete features facilitate editing.

Scripts can also be loaded from or saved to a file.

Having a return value, some events can affect the process flow of further procedure. In the BeforeForward event, you can use the return value to control whether data are forwarded instantly or corrected beforehand.

Furthermore, client events can return notices or other messages to enaio® client and indicate if notifications are depicted in message boxes.

Event Filing

Event code will be saved in the database in the following table.

oswfeventtypes	Here, all event types (e.g. BeforeForward, StartActivity...) with a type ID are defined.
oswfevents	Defines the event type for a model or activity. Each event receives its own ID. Some event types can be defined multiple times in a model/activity (e.g. ButtonClick). The column 'Params' gathers information ensuring an event's uniqueness (e.g. name of the button).
oswfscripts	This table comprises the actual scripts
oswfevtscrrel	Contains relations between events (oswfevents) and scripts (oswfscripts). A script can therewith be connected to multiple events.

Debugging

Users with adequate system roles can activate the debug mode for workflow events in enaio® client. Events are not executed in debug mode; instead the event editor will open the script of the respective activity. The script can there be processed step by step. Current variable values will be displayed and values can be changed.

The script cannot be edited in debug mode.

Examples for Event Codes

In the following, examples are listed in which model-specific actions are often executed with event code. These examples describe general proceeding. The added code fragments will just describe the method, but cannot be executed as, due to overview reasons, they are done without initialization functions and declarations.

Access to the COM Interface of the Client

From a client-side workflow event, you can access the COM interface of enaio® client via the Application object. This object is always available in the client-side workflow event and does not have to be created manually.

```
Application.getobjectname oid, oname
```

Use the `CreateObject` function to access the COM interface of the enaio® client from any script code. After the object is provided, functions to access the DMS can be used.

```
On Error Resume Next
Dim objax
Set objax = Application
If objax Is Nothing Then
    'Object not available
End If
```

Important: In a client-side workflow event, you must not use the `CreateObject` function to access the object `optimal_AS.application`; otherwise the script cannot be debugged.

Access to the Function Library `oxvbbs.dll`

The function library `oxvbbs.dll` encapsulates frequently used functions to access the client's COM interface and facilitates the creation of applications. Routines from this library are often used in VB example codes.

Presetting Process Variables when Starting a Process

This example involves process variables being initialized by assigning values to them before the first activity starts. Values, for example, could be read out of an external file or database, or the point of time for process start could be transferred to a variable.

In this case, a server-side `StartActivity` event is determined for the start activity. The example assumes that a string variable must be allocated in order to record the current start time. Alternatively, variables can also be initialized by preset values. But this can only be realized with fixed variable values. Use a script for values that must be calculated beforehand.

```
Dim sVarValue
'reading access
sVarValue = VarName.value
sVarValue = "hello world"
'writing access
VarName.value = sVarValue
```

`VarName` is the name of the variable. In server scripts, process variables are referenced directly with their names. In client scripts, the support object `wfVariables` enables references. This support object is a collection of all workflow variables.

References to single variables are performed as follows:

```
Dim sVarValue
'reading access
sVarValue = VarName.value
sVarValue = "hello world"
'writing access
VarName.value = sVarValue
```

Starting an Application with an Activity Form

If designated in the process, a 'ButtonClick' event can launch an application at a specific activity, for example, in order to display a document in the workflow file or to query a third-party system.

Adding a Document to the Workflow File

If an activity requires a document to be added to the workflow file for process processing, e.g. in order to create a writing concerning the business case, the COM interface of enaio® client can first add the document to the DMS and then insert it into the workflow file.

Such scripts are usually assigned to the 'ButtonClick' event. For this purpose, the activity form must contain a respective button.

At first, create an instance of the enaio® client COM interface. Then set up a new document in the DMS. The COM programming reference describes how documents are set up. The object will then be inserted into the workflow file. This is realized through the already pre-initialized variable `wfFile`, its method `wfFile.addfileobject` is an `IWFFFile` collection enabling the workflow file access. The appendix offers a detailed description in the workflow programming reference. After adding document, the data sheet and the document will open.

```
'Creating an instance of the OS:Client COM interface
Dim myax
stmpfile = 'Location of the transferred file
Set myax = CreateObject("oxvbscript.function")
'... Initialization of the new document ...
'here the new document is inserted into the DMS
myax.InsertIntoDocument stmpfile, newid, newtype
'Inserting the document into the workflow file
wfFile.AddFileObject newid, newtype, True, 2
Dim mxax
Set mxax = Application
'Open document data sheet
mxax.OpenDataDlg newid, newtype, 1
'Open document for editing
mxax.OpenObjectId newid, newtype, 1
```

Validity Check before Forwarding

The event 'BeforeForward' allows you to perform a validity check before forwarding. This event returns a value to enaio® client that decides if either the activity is ended or a user has to edit entered data.

In the following code fragment, it is checked whether the inserted purchase order value is greater than 0 and less than € 10000. If this is not the case, a message box appears asking you to change inserted data.

```
If wfVariables.Value.Value > 10000 Then
    scriptresult.Resultcode = -1
    scriptresult.ResultString = "Articles can be ordered up to a value of max. 10000
€."
End If
If wfVariables.Value.Value <= 0 Then
    scriptresult.Resultcode = -1
    scriptresult.ResultString = "Please enter a valid value."
End If
If scriptresult.ResultCode = -1 Then
    scriptresult.ShowResultString = True
End If
```

In the next example, it is checked whether at least one document of the type 'Business case' has been created in the activity. If not, a message box will open and the forwarding process will be canceled.

```
doctype = 131072
docname = "Business case"
Set fileobjs = wfFile.fileobjects
bFound = False
For i = 0 To fileobjs.count-1
    Set obj = fileobjs.item(i)
    If obj.objecttype = doctype And obj.isworkspaceobject Then
        bFound = True
    End If
Next
If bfound = False Then
```

```

        scriptresult.ResultString = "The workspace of the file has to contain at least
one object of the type " & docname & "."
        scriptresult.ResultCode = -1
End If
If scriptresult.ResultCode = -1 Then
    scriptresult.ShowResultString = True
End If

```

Transferring Objects to the DMS

Using script code, objects, which so far were only located in the workflow file, can be transferred to the DMS. These objects are thus retrievable with a regular search in enaio® client.

```

Dim myax
Set myax = Application
'Determine object of the type Business case
For lq = 0 To wffile.fileobjects.count -1
    lidtmp = wffile.fileobjects(lq).ID
    lTypetmp = wffile.fileobjects(lq).objecttype
    myax.getobjectname lidtmp, lnametmp
    If lcase(lnametmp) = "Business case" Then
        lidabl = wffile.fileobjects(lq).ID
        lTypeabl = wffile.fileobjects(lq).objecttype
    End If
Next
'Assign FileID and RegisterID
ordid = 4711
regid = 10234
'Move object to the DMS
lRet = myax.moveobject(lidabl, lTypeabl, ordid, regid)

```

Searching in the DMS

Before connecting the DMS and the workflow file in this way, respective objects must be identified in the DMS. The following sample code is a document type query. Thereby, myax references to an object of the client's COM interface.

```

Dim myax
Set myax = Application
oid = 62
sDocName = "Document type"
myax.getobjectname oid, oname
TmpDateiName = "c:\temp\hallo.txt"
With oxHelp
    .WriteProfString "QUERY", "CABINET", oname, TmpDateiName
    .WriteProfString "QUERY", "CABINET ID", oid, TmpDateiName
    .WriteProfString "QUERY", "DOCUMENT", sDocName, TmpDateiName
    .WriteProfString "QUERY", "FIELD1", "holgi", TmpDateiName
End With
sRet=Myax.StartDocRequest(TmpDateiName)
If Trim(sRet) <> vbNullString Then
    If Left(sRet, 5) <> "ERROR" Then
        Set FSO = CreateObject("Scripting.FileSystemObject")
        Set ResultFile = FSO.OpenTextFile(sRet)
        Do While ResultFile.AtEndOfStream = False
            AktIndex = ResultFile.ReadLine
            If Trim(AktIndex) <> vbNullString Then
                FolderQueries.add AktIndex
            End If
        Loop
        ResultFile.close
    End If

```

End If

Inserting a new Document in the DMS

Use the following code to insert a new document in the DMS. The location (ID of the respective folder) must be determined beforehand.

```
Dim myax
Set myax = Application
oid = 24
sDocName = "Document type"
sWert = "Hello"
myax.getobjectname oid, oname
TmpDateiname = "c:\temp\Insert.txt"
With oxHelp
    .WriteProfString "INSERT", "CABINET", oname , TmpDateiname
    .WriteProfString "INSERT", "CABINET ID", oid , TmpDateiname
    .WriteProfString "INSERT", "DOCUMENT", sDocName , TmpDateiname
    .WriteProfString "INSERT", "FIELD1=Fieldname", sWert , TmpDateiname
End With
Dim objax
set objax = createObject("oxvbscript.function")
lRet=objax.insertintodocument(TmpDateiname,did,dtype)
If lRet<>0 Then
    error while inserting
End If
```

Determining the Subsequent Workflow Participant

Within a workflow process, sometimes the subsequent participant has to be determined dynamically. This may happen in case not only workflow variables but also external data sources determine the subsequent participant. Furthermore, a great number of transition conditions may lead to extraordinary efforts when designing the process model. Script code allows you to dynamically determine the subsequent participant. This is set in the 'StartActivity' event.

In the following script code, the process creator is assigned to as the activity's processor. This is useful when the creator will be notified after the processing of activities or has to make further decisions.

```
'The editor of this activity is the creator of the process
thisProcess.SetActivityPerformer
thisProcess.CurrentActivity.ActivityId,thisProcess.Creator
```

Any other role or person can be alternatively defined as subsequent participant. To do so, the GUID of the intended role or person is specified as parameter.

```
Dim sUserID
sUserID = 'Organizational object ID of the user
thisProcess.SetActivityPerformer thisProcess.CurrentActivity.ActivityId,sUserID
```

Navigating the Organizational Structure

Navigation within the organizational structure can be realized using script code for functions to determine preceding and successive workflow participants. Supportive functions do thereby access names of organizational units and attributes. Navigation results can be used to specify the previous participant.

The following sample code determines whether the current role is assigned to a 'team', and if so, whether there is a 'team spokesperson'. The query is performed using the name of the organizational unit, and depends on the model. In the same manner, other organizational units can be searched.

```
set roles = wforganisation.GetOrgObjectsByClassName("Role")
for i = 0 To roles.count-1
    set role = roles.item(i)
    'The percent sign is transferred to the database and works as a wildcard
```



```

Set teams = role.GetPredecessorsByClassName("Team%", true)
For j = 0 To teams.count-1
    'Determining the team spokesperson
    Set teams1 = teams.item(j).GetSuccessorsByName("%Team spokesperson%")
    If teams1.count >0 Then
        r_ts.name.value = teams1.item(0).name
        r_ts.guid.value = teams1.item(0).id
        bFound = True
        Exit For
    End If
Next
Next

```

Determining all Users and their Roles

In the following example, all users and their roles are determined. Please note that the functions GetXYZByName can also be transferred as wildcards. All objects are determined from which parts of the name match the search string.

```

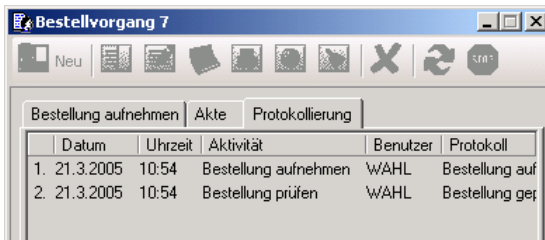
'The percent sign is transferred to the database and works as a wildcard
Set users = wfOrganisation.GetOrgObjectsByClassName("Person%")
For i = 0 To users.count -1
    Set username = users.item(i)
    'A variable, here li_users, of the li_user structure type must exist,
    'to create a list with this structure
    Set UserItem = wfvariables.li_users.CreateListItem()
    UserItem.guid.value = username.id
    UserItem.g_name.value = username.name
    Set roles = username.Predecessors
    For j = 0 To roles.count -1
        set role = roles.item(j)
        'In the structure li_user, a further structure li_roles has to exist,
        'to create a list of this structure as well
        Set RoleItem = UserItem.li_roles.CreateListItem()
        RoleItem.guid.value = role.id
        RoleItem.g_name.value = role.name
        'Name and ID are added to the list li_roles
        itemid = UserItem.li_roles.AddListItem(RoleItem)
    Next
    'Name and ID are added to the list li_users
    itemid = wfvariables.li_users.AddListItem(UserItem)
Next

```

Setting up the Log

If configured accordingly, the process flow so far can be viewed at the processing steps in enaio® client. The logging provides information on processed activities and can be configured for each process model separately.

Both the separate log settings and script code enable the user to customize the log and create relevant entries only.



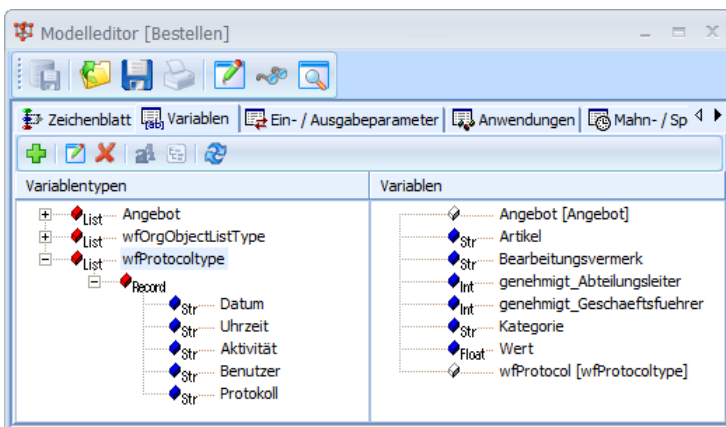
	Datum	Uhrzeit	Aktivität	Benutzer	Protokoll
1.	21.3.2005	10:54	Bestellung aufnehmen	WAHL	Bestellung auf
2.	21.3.2005	10:54	Bestellung prüfen	WAHL	Bestellung ge

Log setup requires multiple steps. At first, a variable type must be defined, and a log variable must then be created that is kept throughout the entire workflow process. Afterwards, this variable must be filled in the BeforeForward event or elsewhere to log the respective activities. Activity logs are displayed automatically.

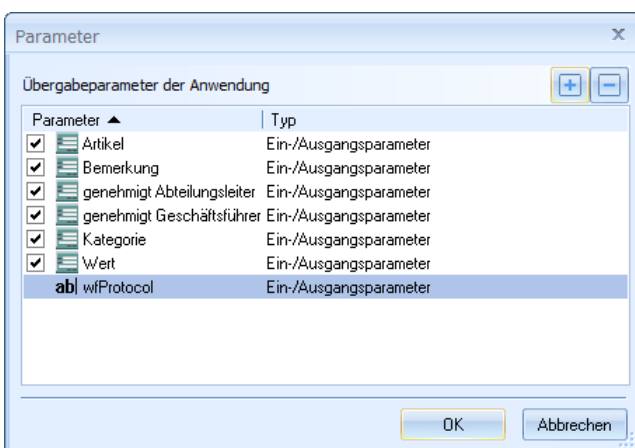
Defining the Log Variable Type

The variable for logging is a combined, custom type. This format has been selected as it allows defining the number and names of columns in the process.

Variable and variable type is predefined for each model.



As soon as the variable has been made available in the process model, it has to be transferred to the individual applications. This is done when configuring the application's transfer parameter. When adding the parameter, enter the name 'WFProtocol' and define it as 'Input and output parameter'. Therewith, log entries of processed activities so far are transferred to the client and can be extended before forwarding.

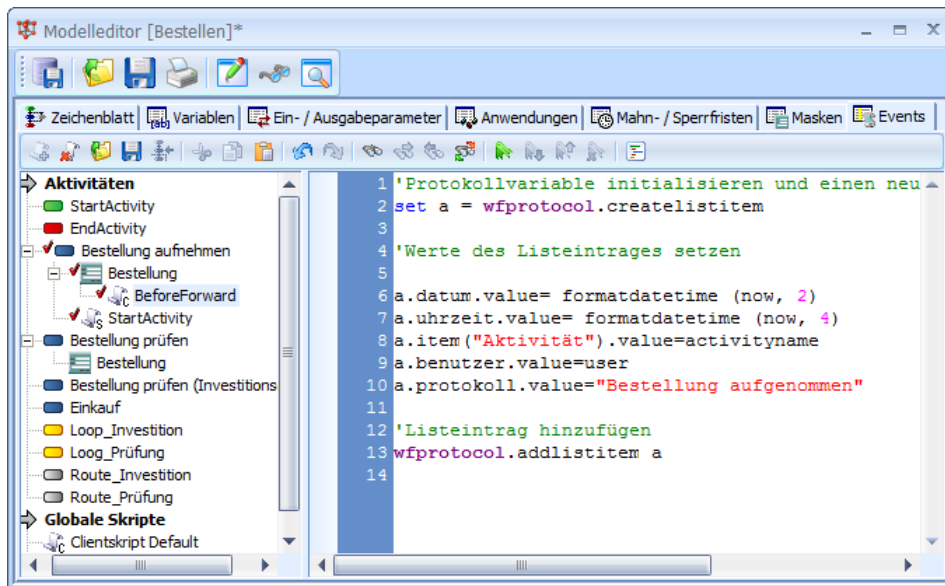


For each activity, the application parameter 'WFProtocol' is assigned to the workflow variable 'WFProtocol'.

Having performed these steps for all applications and activities, the log is available for all processing steps. The next step includes how to fill the log variable with reasonable values.

Creating Log Events

As mentioned above, applications itself have to fill the log variable. This is conveniently done with event code at the entry point 'BeforeForward'. Naturally, this can also be done at other entry points like 'BeforeOpen' or 'ButtonClick' to record users personalizing activities or clicking buttons.



The code is quite simple, and is displayed here again. Please note that this event code must be made available for all activities for which logging must be extended. As a supplement, this code can be added to other scripts. Usually, the supplement is added at the end of a script to not perform invalid loggings in case of script errors.

```

'Initialize log variable and create new list entry
set a = wfprotocol.createlistitem
'Set list entry value
a.datum.value= formatdatetime (now, 2)
a.uhrzeit.value= formatdatetime (now, 4)
a.item("Activity").value=activityname
a.benutzer.value=user
a.protokoll.value="Purchase order taken"
'Add list entry
wfprotocol.addlistitem a

```

Fields are addressed separately according to the defined log variable type. So, additional fields can be addressed and the entry of the 'Log' column formatted further.

As shown here, variables containing umlauts have to be addressed with: `Variable.item(„<Variable name>“).Value`. The VB Script function `formatdatetime` has to be used to set current date or time.

Integrating Digital Signature and Authorizing Steps

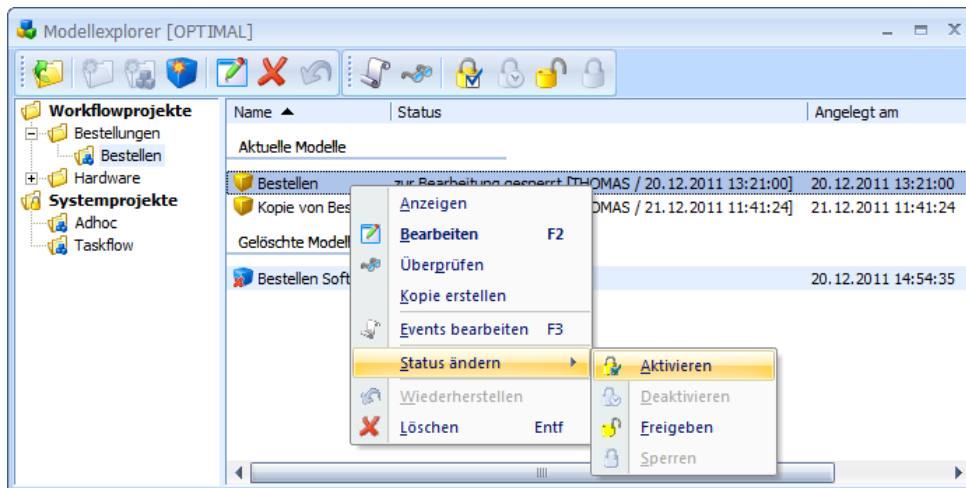
The integration of digital signature allows you to sign documents and to check existing signatures within the workflow. In addition, legal authorization of single process steps and entries is possible.

Apart from that, password dialogs can be deployed for process step authorization. Users must therewith enter their login password into a dialog before forwarding activities.

Set up the authorization type and whether or not to perform authorization in the activities' extended attribute.

Model Activation

Until activation, workflow models pass through different states. This is due to the workflow engine being able to process valid models only and some modeling steps possibly not being completely defined. For that reason, each model has a state that is set up in the workflow editor. At the same time, it ensures that multiple users cannot simultaneously edit a model mutually overwriting their changes.



Following status information types can be set for single models:

Status	Meaning
Lock	Model is locked for editing (checked out by the editor)
Release	Model is released for editing (checked in by the editor)
Use	Model is made available at the server, but new processes cannot be started. Processes that have been started with this model continue until they are finished.
Activate	Model is activated at the server, new processes can be started.

Please note that only one active model can only be available for each workflow family. If multiple dependent process models are administered in a workflow project, the engine deploys only one current model for processing new processes.

When changing model versions, used models can be set to 'in use' and a new model activated. Already started processes remain active with the old model until the end and new processes are started with the currently active model.

You cannot activate models or set them to 'in use' as long as they are syntactically incorrect, i.e. if no applications were assigned to activities, for example. Syntax is checked when switching the status in the workflow engine. If the check fails, the status will not be changed and the user will receive an error message.

Use the workflow administrator to cancel already started processes. This is below described in the chapter *Operation and Administration*.

Please note that only workflow models of active organizations can be used, although workflow models of inactive organizations may receive the status 'active'.

Import and Export of Workflow Projects

The import and export of workflow projects allows you to exchange models, organizational structure, and forms between development, test, and productive systems. Since process model design requires extraordinary efforts, it is necessary to transfer changes and extensions to other target systems without recreating them step by step.

The import and export functions of the workflow editor are available for this purpose. Use the context menu in the workflow editor to import and export organizations, projects, families, and models.

As a rule, configurations are upwards compatible with different enaio® versions, that is, configurations created in earlier system versions can also be used in later system versions.

Downward compatibility is by contrast not guaranteed since enaio® platform developments result in new features and extended configurations. As a consequence, configurations created in later system versions cannot be used in earlier system versions.

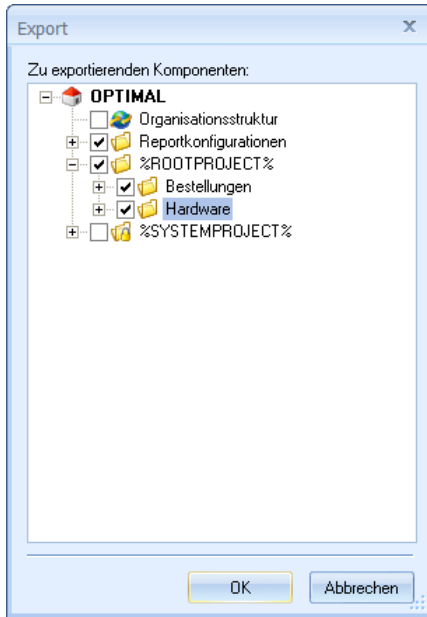
Import attempts of configurations which have been created in earlier system versions, will with the utmost probability result in feature settings getting lost, format incompatibility and/or program errors. Therefore you must pay attention to system versions when dealing with test, development and productive systems and in any case abstain from attempting to import and activate configurations which have been created in later system versions. This applies to configuration files, in particular to object definitions and workflow models.

Models must be exchanged only between systems with the same version. Exchanging models between different system versions requires detailed tests to be performed.

Export

Components assigned to an organization are exported. The export dialog lists the organizational structure and all assigned projects, families, models, and report configurations.

Data of all selected objects are exported into an XML file.



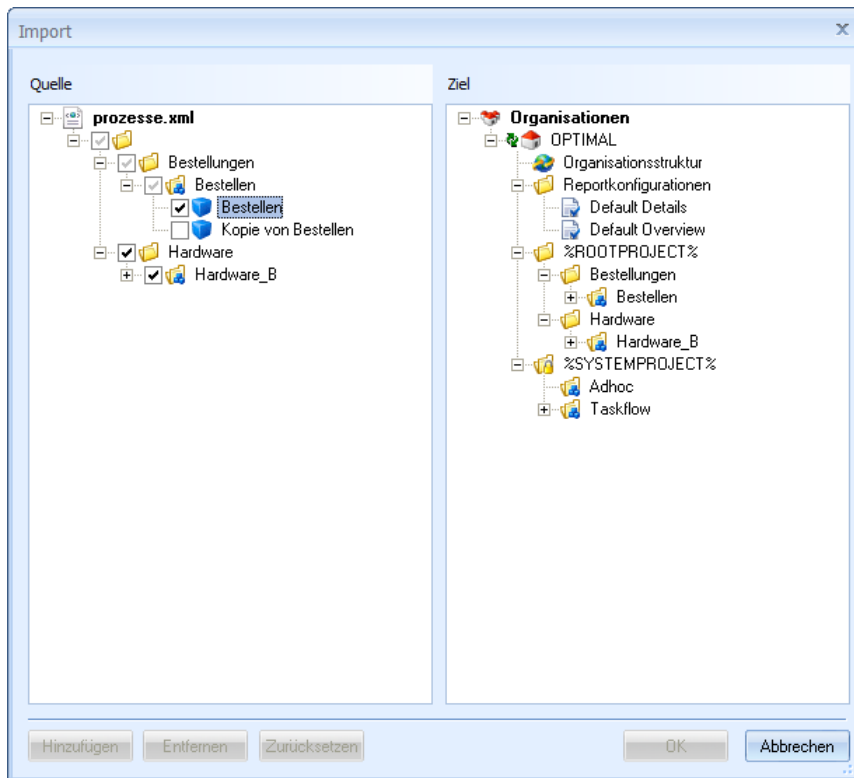
It is also possible to export a single model opened in the model editor into an XML file.

Import

For import, organizational structure, report configuration, projects, families, or models are assigned to an already set up organization.

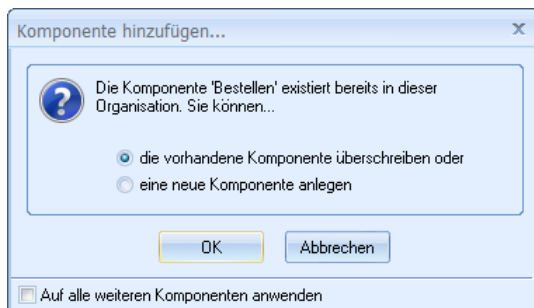
The import dialog lists all objects contained in the selected XML export file and all set up organizations, projects, families, and models.

Select objects of the XML export file and drag and drop them onto the respective organization level.



All imported models are set as 'available for editing'.

In case there already are components with the same GUID, a confirmation dialog will open.



You can choose to replace existing components or to provide imported components with new GUIDs. Names will not be changed.

Open a model in the model editor to import data from an XML file that have been exported with the model editor before.

Standard Ad Hoc Workflow And Taskflow

So that you can access the workflow functionality quickly and easily, enaio® provides you with models that allow you to get a productive system with workflow functions up and running in minutes, without the need to design a complex model yourself. Two models can be integrated with just a few adjustments:

- Taskflow

In every-day business, many processes mainly consist of one step only: delegation of tasks. In enaio® client provides many ways to execute a task – with taskflow workflow advantages are additionally available, for example substitution, personalization and logging.

- Standard Ad Hoc Workflow

This model includes frequent process steps that company processes consist of, such as acknowledgements, releases, polls, inquiry, collaboration, or review.

Standard Ad Hoc Workflow

This model includes frequent process steps that company processes consist of, such as acknowledgements, releases, polls, inquiry, collaboration, or review.

The standard ad hoc model will considerably enhance productivity and process security in OS|ECM systems. At the same time, it allows you to evaluate all possibilities when introducing standardized workflow processes to digitally visualize existing, unofficial processes without fixing the user's business processes in inflexible structures. Step by step, constantly recurring sub-processes can first be saved in circulation slips and then be transferred to defined models.

The standard ad-hoc model is designed for extremely common application cases and does not use all the technical features of enaio® Workflow. The model is not intended to be modified by the user in order to ensure broad applicability and to keep maintenance simple. However, a similar model can be developed by the user or within a project to implement specific requirements in more detail.

Installation

During the installation of enaio®, the workflow model `osdefaultmodels.xml` is copied into the directory `\clients\admin\`. Use enaio® editor-for-workflow to import the model that contains all data necessary for the standard ad-hoc workflow and the taskflow.

Make sure that the WF jobs `WFM::WorkItemNoti`, `WFM::WorkerJob`, `WFM::SpoolerJob`, and `WFM::CheckJob` are activated in enaio® enterprise-manager.

In addition to this model, an object definition file for the standard ad-hoc workflow is available for setting up a cabinet, a register type, and document types in order to automatically file log files of ad-hoc processes in enaio®. Use a configuration file to control this log administration.

Log files can also be created as image files and saved to the filing tray of the process supervisor. In such case, a log configuration is not required.

Configuration

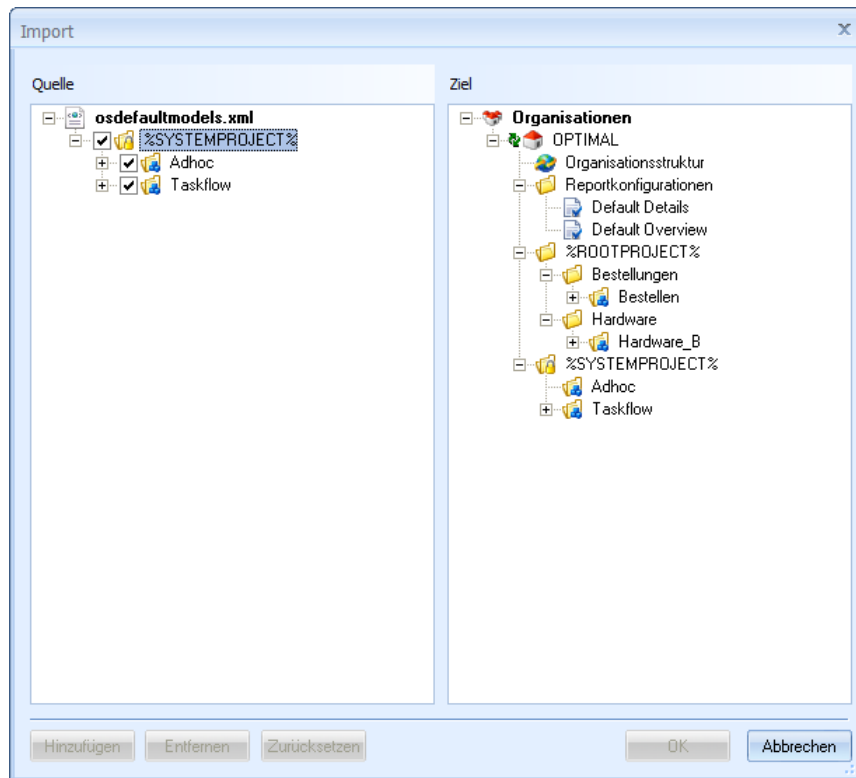
The standard ad hoc model is configured in three steps:

- Import and activation of the model `osdefaultmodel.xml`.
- User assignment using the organizational model.
- Setup of object types for logging and configuration of the log filing (optional).

Importing the Workflow Model

Import the workflow model via enaio® editor-for-workflow.

- Start enaio® editor-for-workflow.
- Select the function **IMPORT ORGANIZATION** from the toolbar or the **FILE** menu.
- Use the file selection dialog to open the file `osdefaultmodels.xml` located in the directory `\clients\admin`.
- Select the entry 'System projects' in the area **SOURCE** and the entry 'Organization' in the area **TARGET**.
Alternatively drag and drop the entry 'System projects' onto the entry 'Organization'.

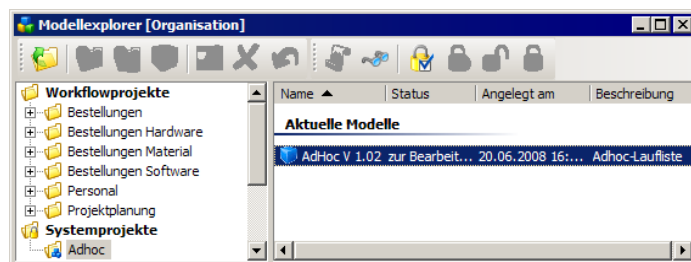


Click the **ADD** button. The project will be imported and the dialog will close.

Instead of importing the three language versions of the standard ad-hoc model and the taskflow, it is possible to import only the one model that you intend to use. The selected language version sets only the language which is used by enaio® server for communication and logging.

Independently of the selected language for server communication and logging, the workflow steps are available in each of the three languages: German, English and French.

- Open the model explorer. The new system project will be displayed.



The imported models are write-protected and cannot be edited.

- Select the intended language version of the model and activate it through the respective function in the context menu or the toolbar.

Automatic check will inform you about no document supervisor being entered. Events ensure that this function is automatically assigned to the user that starts a process. After the end of a process, all documents in the workflow file not having any location will be transferred into the personal filing tray.

Therewith, the model is integrated and active. In the following, roles and persons are bind in.

Configuring the Organizational Structure

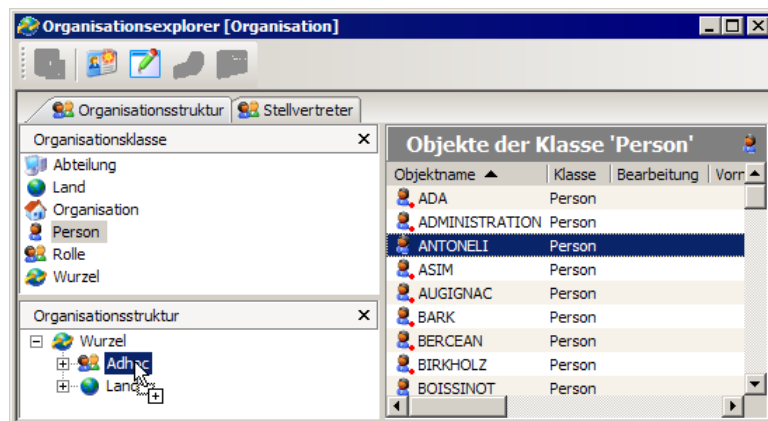
When installing enaio®, the role 'Ad hoc' is automatically created on the top level of the organizational structure. All persons that you define as subobjects of the 'Ad hoc' role in the organizational structure can start ad hoc processes and taskflows in enaio® client, given that they are provided with the system role 'Client: Use workflow'.

Other persons and roles of the organizational structure can be entered as process participants when configuring the circulation slip of an ad hoc process.

Use enaio® editor-for-workflow to edit the organizational structure.

- Start enaio® editor-for-workflow.
- Open the organization explorer from the workspace.

- Assign the objects of the class 'Person' to the role 'Ad hoc'.



- Save all changes.

Ad hoc processes can now be started and executed in enaio® client.

After user import into the organization explorer, assign the role 'Ad hoc' to them.

You can optionally create and integrate additional roles for ad hoc processes. Events from the context of running processes assign users that had played a role within the process flow to these roles.

The following roles are available:

Role	Meaning
%Erste Empfänger der Aktivität	If a circulation slip contains an activity more than once, this role determines all participants (roles and persons) that have been assigned to activity the first time.
%Erste Bearbeiter der Aktivität	If a circulation slip contains an activity more than once, this role determines all participants (roles and persons) that have performed the activity the first time.
%Vorherige Empfänger der Aktivität	If a circulation slip contains an activity more than once, this role determines all participants (roles and persons) that have previously been assigned to the activity.
%Vorherige Bearbeiter der Aktivität	If a circulation slip contains an activity more than once, this role determines all participants (roles and persons) that have previously performed the activity.
%Letzte Bearbeiter	This role determines the participant that has forwarded a process step at last.
%Prozessinitiator	This role determines the participant that has started the process.

Use the organization explorer to create these roles as objects of the class 'Role' assigning them to the organizational structure.

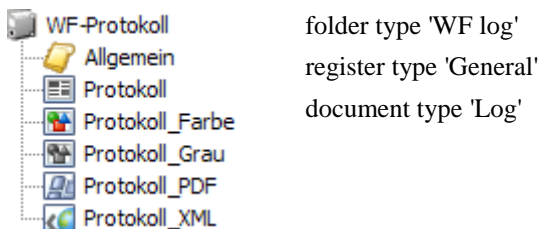
After creation, these roles might be available to users in other workflow processes as well. But will not have any function as no event embeds them.

Logging Configuration

The activity 'Log' is an event-controlled part of the model 'Ad hoc' and collects all data of already completed actions in a log. As part of a circulation slip it is performed automatically. The log is afterwards located as a document in the workflow file. In the following activity, assign the log to a document type, index and save it. If this is not done, the log is filed automatically into the filing tray of the user that has started the process.

Specify the log format and configure the location that the log is saved to using a configuration. The configuration is optional, without it the log is filed as an image document into the workflow tray.

The object definition file defines the following object types for the ad hoc workflow logging:



The configuration file is synchronized with this object definition. A folder is created for each year, a register for each month and therein a document containing the logged data.

The document's index data include the process name, the process ID, and the subject of the process step:

Importing the Object Definition

Launch enaio® editor and open the object definition file AdHoc_asobjdef.xml located in the directory \clients\admin\.

Import the cabinet, the register, and at least one document type into the system object definition. Only the document type corresponding to the intended log format is required.

Save the object definition and adapt the tables.

Information on enaio® editor can be found in the respective handbook.

The object types' IDs are required for further configuration. These IDs can be found in the column 'Object type' of the property dialog.

Example:

The document type 'Log' has the ID 131115

Datenbank	
Bezeichnung	Protokoll
Tabellenname	object103
Objektyp	131115 / 2 / 43
interner Name	WF_Protokoll_TIF

You must provide users having the role 'Ad hoc' with respective access rights in the security system so they can create configured folders, registers, and documents for logging.

Log Configuration File

Specify the log's file format and its filing location in the log configuration file AdHoc_Config.xml.

The file is installed in the \etc directory of your enaio® installation's data directory. If you do not change this file, log files are saved as PDFs. If you delete the configuration file, only an image file without document type is created.

Log creation in PDF format requires a JRE version 1.3 or later on each enaio® server, the JAVA_HOME environment variable to be set to the Java installations path ... \java\jdk\, and activated conversion engines.

The log configuration file comprises the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<Protocol Type="NONE">
```

Enter 'XML', 'TIF', 'PDF', or 'NONE' as log type. Enter 'NONE' in case you just want a log in the file without automatic transfer to a location. In this case, only the document type for the log file will be transferred from the following configuration.

Enter 'TIF' for an image document. The file format for filing (JPG or TIFF) is defined automatically through the assignment to an image document type. Enter 'XML' for an XML document.

```
<Folder ObjectId="0" InternalName="WF_Protokoll">
```

Enter the folder ID and the internal folder type name.

```
<Fields>
<Family InternalName="WF_Family"/>
<Year InternalName="WF_Year"/>
</Fields>
```

Enter the internal field labels into the folder fields to which the model name and the year are assigned.

```
</Folder>
<Register ObjectId="6488064" InternalName="WF_General">
```

Enter the register type ID and the internal register type name.

```
<Fields>
<Month InternalName="WF_Month"/>
</Fields>
```

Enter the internal field label into the register field to which the month is assigned.

```
</Register>
<Document ObjectId="458752" InternalName="WF_Protocol_PDF">
```

Enter the document type ID and the internal document type name which you have set up for the intended file format in the object definition.

```
<Fields>
<ProcessName InternalName="WF_ProcessName_PDF"/>
<ProcessId InternalName="WF_ProcessId_PDF"/>
<Comment InternalName="WF_Comment_PDF"/>
</Fields>
```

Enter the internal field labels into the document type fields to which the process name, the process ID, and the subject are assigned.

```
</Document>
</Protocol>
```

You can also adjust this configuration to other archive object types. A structure with folder type, register type, and document type is required. The object type ID and the internal name are always used for assignment.

If you select the XML format for logging and the assigned document type 'Log_XML,' you can assign the style sheet `adhoc.xml`, located in the `\etc\templates\` directory of the data directory, to the document type for the view in enaio® client by entering it into the `as.cfg` file located in the `\etc\` directory of the data directory.

When updating enaio®, the configuration file `AdHoc_Config.xml` in the `\etc` directory of the data directory, the style sheets `adhoc.xml` and `adhocFO.xml` as well as their equivalents in other languages in the `\server\etc\templates` directory will not be replaced. This way it is ensured that updates do not undo customizations. Nevertheless, updates will always replace the default files (`AdHoc_Config_default.xml`, `adhoc_default.xml`, `adhocFO_default.xml` etc.) which are also found in the directories `\etc` and `\server\etc\templates` of the data directory, thus always serving as up-to-date templates for customizations.

Information on XML document types can be found in the enaio® administrator handbook.

Standard Ad Hoc Workflow Sequence

The standard ad hoc workflow offers modeled processes in which users can work in simply and quickly and which are still complex enough to organize most of the daily recurring working processes.

The standard ad hoc workflow consists of an area with seven well-matched activities that – unlike in the structured workflow – are not connected through predefined transitions but with which a flexible process flow is created by users during execution. Three different processes are performed with these activities: poll processes, release processes, and information processes. You can arbitrarily combine these processes and deploy them more than once. A process is completed once the circulation slip does not contain any activities.

The standard ad hoc workflow file can be processed by all participants in all steps. The **LOGGING** tab in each step allows you to track the technical process flow. Each step offers an area in which users can enter information to create an annotation history. It can be viewed by all participants in all steps on the respective tab.

The standard ad-hoc workflow can be executed in both enaio® client and enaio® webclient. All steps are available in German, English, and French.

In the following, single activities and their relations to preceding or subsequent activities are described.

A general description of the workflow, the workflow file, and circulation slips can be found in the enaio® client handbook.

Initialization

An ad-hoc process is started from the 'Startable processes' inbox in enaio® client. The person who starts the process becomes the initiator. If a process is finished and there are still documents and logs in the workflow file that do not have a location in enaio®, they will be placed in the initiator's personal filing tray.

The first step of a started process is always the 'Initialization' step.

The workflow form of the process step offers three editable areas on the **GENERAL** tab:

- **SUBJECT**

The **SUBJECT** field is a required field: it must be filled in. The subject indicates all process steps in the inbox and can be used for the indexing of log files. That is why it is recommended to enter a reasonable description for the process.

The entry of the **SUBJECT** field can only be modified in a subsequent 'Initialization' step.

- **COMMENT**

The comment field will be displayed during all steps and can be modified only in a subsequent 'Initialization' step. Its entry will be displayed in all following steps and also be entered into the annotation history.

- **POLL**

With the **POLL** field you can specify the poll options of a following 'Poll' step in a table. Poll options can be modified in a subsequent 'Editing' or 'Initialization' step. When changing the poll or its options, already existing poll results will always be reset.

The poll options entered here will be listed in the 'Poll' step and participants can select the requested option. The poll option 'No opinion' will be added automatically.

As a multi-instance activity, a poll can be executed by more than one participant. The poll will be finished automatically if either all participants cast their votes or if any poll option exceeds a specified threshold.

The poll result will be displayed in a subsequent 'Notice' step and on the **LOGGING** tab.

The **CIRCULATION SLIP REMARK** field shows the comment entered for this activity into the circulation slip.

1. Abstimmung

Circulation slip remarks are optional. In the first initialization step, the comment field will still be empty.

When forwarding a process step, users will be notified if the workflow file does not contain any object.

Except an additional initialization, the **ANNOTATION HISTORY** tab of all following steps displays the participant and the process initialization date in the **INITIATOR** and **START DATE** fields.

The annotation history on the respective tab is still empty in the first process step.

Revision

In the 'Revision' step, the workflow form offers the fields **SUBJECT** and **COMMENT** containing the data entered during initialization and the field **CIRCULATION SLIP REMARK** into which the comment for this activity was entered. These three fields are write-protected.

Entries of the **NOTE** field will be entered into the annotation history and are visible in all subsequent process steps.

As for initialization, you can specify or edit the poll options in a table on the **EDIT POLL OPTIONS** tab. Already existing poll results will be deleted and can be accessed only if they were saved during a preceding logging step.

Collaboration/Query

In the 'Collaboration/Query' step, the workflow form also offers the write-protected fields **SUBJECT** and **COMMENT** containing the data entered during initialization and the field **CIRCULATION SLIP REMARK** into which the comment for this activity was entered.

Entries of the **RESPONSE** field will be entered into the annotation history and are visible in all subsequent process steps.

Poll

The workflow form of the 'Poll' step lists the poll options specified in a table during the initialization or editing step.

Beyond the specified poll options, participants have always the possibility of selecting the 'No opinion' option. The **POLL** field is a required field: it must be filled in.

As a multi-instance activity, a poll can be executed by more than one participant.

The poll will be finished automatically if either all participants cast their votes or if any poll option exceeds a specified threshold.

The poll result will be displayed as a table in a subsequent 'Notice' step and on the **POLL RESULT** tab. Use the 'Logging' tab to view the individual vote of each participant. The logging table can be copied to the clipboard through the context menu or **CTRL+E**.

The number of 'No opinion' votes will not be displayed in the table.

Release

The fields **SUBJECT** and **COMMENT** contain the data entered during initialization and the comment for this activity was entered into the **CIRCULATION SLIP REMARK** field.

In the 'Release' step, you can select an option from the **DECISION** area – **OPEN**, **NO RELEASE** or **RELEASE PERMITTED**.

The selected option will be logged and can be viewed during all subsequent steps on the **LOGGING** tab.

If a further 'Release' step follows, the option selected before will be preset.

Entries of the **NOTE** field will be entered into the annotation history and are visible in all subsequent process steps.

Notice

As well in the 'Notice' step, the workflow form offers the write-protected fields **SUBJECT** and **COMMENT** containing the data entered during initialization and the field **CIRCULATION SLIP REMARK** into which the comment for this activity was entered.

Entries of the **NOTE** field will be entered into the annotation history and are visible in all subsequent process steps.

The result of a previously executed poll will be displayed as a table on the **POLL RESULT** tab. Use the 'Logging' tab to view the individual vote of each participant.

Release information can be found on the **LOGGING** tab.

Log

The 'Log' step has no participants. Thus, it is not started by any participant but performed automatically. As part of the circulation slip, this step creates a file in which all data of performed steps so far are logged and that is filed into the workflow file. The log file can be viewed during all following steps.

The 'Log' step can be at the end of a process. If there is no next step, a log file to which a location and document type was assigned in a configuration is moved from the workflow file to the initiator's filing tray in common with all documents without location.

Beside this logging process, a technical trace logging is automatically performed on the **LOGGING** tab. It logs, for example, release information, but not notes of single steps. Notes can be viewed in the annotation history.

The **LOGGING** tab's content can be copied to the clipboard with the context menu or **CTRL+E**.

Escalation for Standard Ad Hoc Activities

An escalation set up in the model can be assigned to an activity in a circulation slip. The following escalations are available for all activities of the standard ad hoc workflow:

- reminder e-mail
After not having forwarded a process step within a certain period of time, the user will receive an e-mail with a respective notice. The period of time is indicated in the fields **DUE ON** and **AT**.
This is subject to the e-mail address being specified in the workflow user administration.
- automatic forwarding
The process step is automatically forwarded to the following step as soon as the end of the time period is reached. Both personalized and not personalized steps will be forwarded. The preset time period (two days) can be changed in the fields **DUE ON** and **AT**.
- sending to substitute
After not having forwarded a process step within a certain period of time, it will be sent automatically to the substitute. The period of time is indicated in the fields **DUE ON** and **AT**.
This is subject to a substitute being specified in the workflow user administration.

Taskflow

The standard ad hoc workflow may be used for quick task assignment, but due to its complex circulation slip management possibilities users may find it not easy to manage.

The taskflow by contrast consists of only one step which can be executed and routed to team members for resolution. The recipient will then process the step. The process will be forwarded and routed back to the sender. The recipient can include further recipients and will thus become the sender or send the process back.

Installation and Configuration

Together with the default ad hoc workflow, the taskflow is part of the `osdefaultmodels.xml` file in the `...clients\admin` directory and it is installed with **OS|EDITOR_for Workflow** by importing the file (see 'Installation').

As is the case for default ad hoc workflows, users also have to be assigned the 'Adhoc' role through the organization model. Every user with this role will become taskflow participant with equal rights.

Taskflow Sequence

The taskflow consists of one process step.

For a process step a task can be defined and detailed with options, such as **for editing**, **for checking**, **for approval**, and **for notice**. In addition, a due date which in the model corresponds to a dunning period, can be defined.

You can select every user as recipient who is authorized within the model. Select the recipient through the add-on.

You can design a process so that a process step has to be read, checked or approved by a recipient and, if necessary, is forwarded to another recipient.

Every process step can be commented on by the recipient. The comments are automatically logged in the history section.

Every process step is automatically returned to the sender and to the initiator eventually so that the initiator is always informed about the task's completion.

Operation and Administration

Licensing and Rights

enaio® workflow is part of enaio® and requires enaio® client to have been installed. All modules use different license keys which are listed below.

License key	Meaning	Checked by
ASW	start of the workflow engine	workflow engine in the application server (oxjobwfm.dll)
WFA	start of the workflow administrator	workflow administrator (axwfadm.exe)
WFE	workflow editor: organizational structure	workflow editor (axwfedit.exe)
WFG	workflow editor: form and process design These actions are possible without 'WFG': <ul style="list-style-type: none"> importing, creating, editing and deleting projects and families, renaming, deleting and changing the status of models, but no locking. 	workflow editor (axwfedit.exe)
MWC	workflow client	enaio® client (ax.exe)
AVG	automatic process start	workflow start component in enaio® capture (axwfstart.exe) and as separate program (axwfscript.exe)
WAU	start the workflow autostation	workflow autostation
WAH	ad hoc workflow	ad hoc workflow

Rights

Administration and process design of a workflow management system must be protected from being accessed by unauthorized persons as such a system may be deployed in sensitive company areas.

For that reason, anyone using the workflow administrator or the workflow editor must be provided with respective system roles.

The following system roles concern the workflow:

- ☐ OSIECM - WF-Admin: Starten
- ☐ OSIECM - WF-Editor: Starten
 - ☐ OSIECM - WF-Editor: Organisation bearbeiten
 - ☐ OSIECM - WF-Editor: Benutzer an-/abwesend melden
 - ☐ OSIECM - WF-Editor: Modell erstellen
- ☐ OSIECM - WF-Prozesse: Per Import starten
- ☐ OSIECM - WF-Simulation: Starten
- ☐ OSIECM - WF-Script: Starten

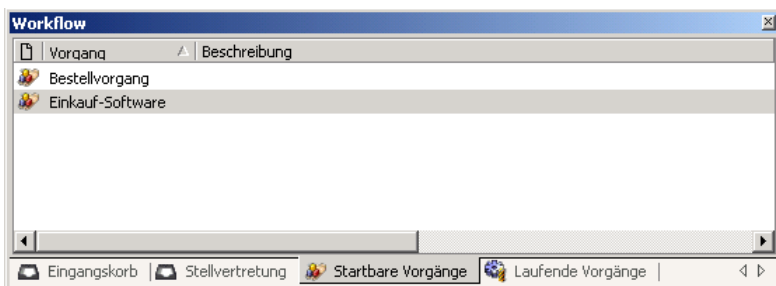
Workflow Components in enaio® client

The most important applications for using the workflow are enaio® client and enaio® webclient. All interactive input for editing the activities of the workflow process takes place in these clients.

enaio® client and enaio® webclient are structured in the same way.

Workflow Area

The workflow area in enaio® client is organized similarly to the subscription and follow-up window. The window can be docked and moved in any way by the user whereby the last setting is saved.



The workflow area is divided into multiple inboxes which can be configured by the user. Processes, activities or models can be administered in these inboxes.

The most important inboxes are the (main) inbox and the list of startable processes.

Inbox


The inbox informs the user about upcoming activities. If the workflow engine has recognized an activity which is supposed to be processed by users of a particular role, the role is resolved for the respective users. All users being part of the role will receive a notification about the presence of a new activity which will be displayed in the inbox.



Inbox icons indicate whether activities have been personalized or whether a set dunning period has been exceeded.

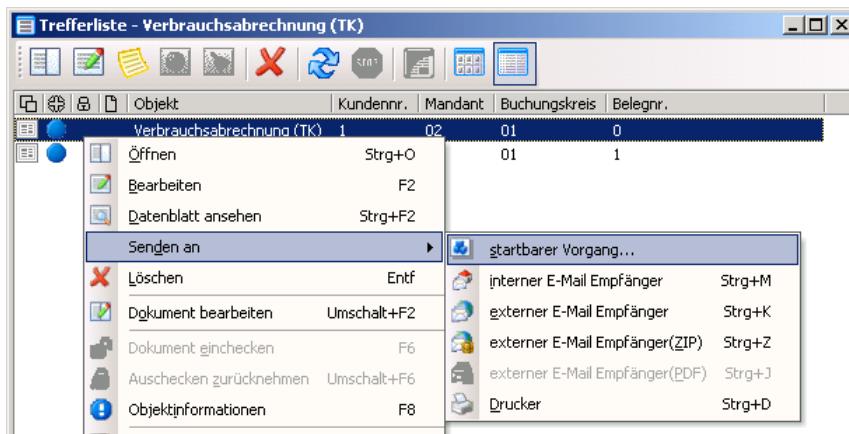
An e-mail notification can also be set up in enaio® enterprise-manager and configured for each activity.

Personalization

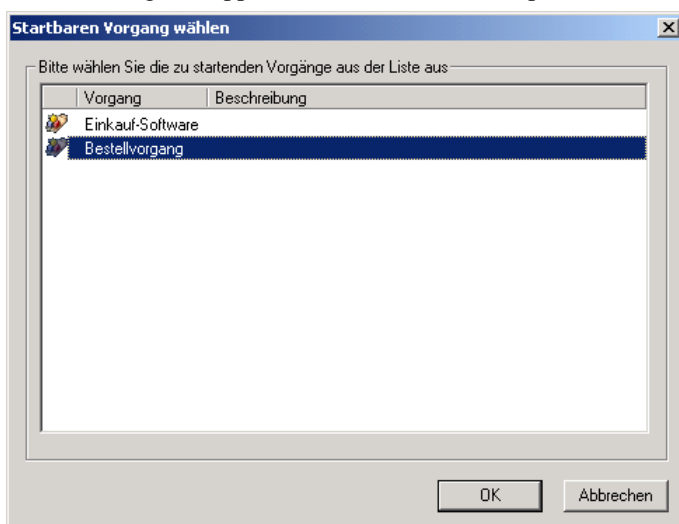
After the first user has accepted the activity by double-clicking on it in the inbox it becomes personalized. Thus, the activity disappears from the inboxes of the other role members and is flagged with the  icon in the inbox of the editor. Usually the activity form predefined in the model appears and the user can execute the scheduled actions.

Startable Processes

The list of startable processes contains all activated models which can be started by the user i.e. by the role which he is part of. Usually, these are started by double-clicking the corresponding entry. If it was set in the model that the process can (or must) be started with a document or any DMS-object, the process can be initialized by dragging the object onto the startable process. Alternatively the **SEND TO...** menu in the Client contains the **STARTABLE PROCESS** entry.



Thus, a dialog will appear in which all 'Startable processes' are listed.



If a process has been launched, the corresponding processing step is entered in the inbox of all participants.

The processing step can be opened via an instant notification if the person who started the process is also participant of the first processing step. Instant notifications are enabled with the activity's notification dialog.

This option requires the specification of a time period in the file `as.cfg` located in the directory `\etc` of the data directory in which the instant notification is sent:

```
[WORKFLOW]
```

```
AUTOSTARTWAITTIME=time period in seconds
```

The period must allow the server to process all data and the user to process and assign the activities. In most cases, a period of 20 seconds will be sufficient. After this period has expired, a process step will no longer open automatically.

Instant notification will also appear if a user has forwarded a process step to the following one he is participant of as well.

Activity Form

Activity forms which are preset by the process design are used to edit activities. The user will enter the required data for each single step.

Bestellvorgang 2

Bestellung aufnehmen | Akte | Protokollierung

Artikel
Netzwerkkarte

Kategorie
Hardware

Wert(EUR)
26.0

☐ genehmigt Abteilungsleiter
☐ genehmigt Geschäftsführer

Bemerkungen
für intern

Weiterleiten | Schließen

Bereit 0 0

The functions of the dialog elements correspond with those on the indexing form in enaio® client. Users can finish an activity by **Forwarding** it. A field on a form can be linked to the workflow-recipient add-on. If it is linked to a following activity, all persons or roles selected using this field will become additional participants of this activity.

If an activity is closed, changes can be applied or discarded.

After forwarding, the process step can be opened via an instant notification if the person who forwarded the process is also participant of the next processing step.

Running Processes

Processes remain visible in enaio® client for users who have started these processes. In the workflow area, the **Running processes** tab displays current process steps and, if personalized, the users. The file of a process step can be viewed. Process steps can be assigned to other users.

If persons marked as supervisors for a model are indicated they can also view the activities on the tab.

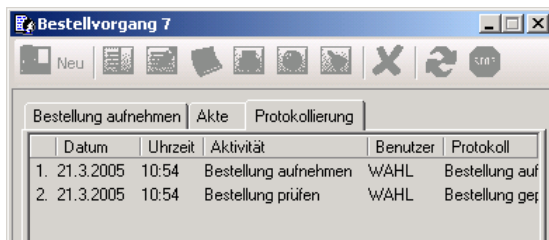
Vorgangsschritt	Vorgang	Benutzer	gestartet am	fällig am
Bestellung prüfen	Bestellvorgang 7		21.03.2005 10:54:14	
Bestellung aufnehmen	Bestellvorgang 8		21.03.2005 10:55:49	23.03.2005 10:55:50
Bestellung aufnehmen	Bestellvorgang 9	Wahl	21.03.2005 10:55:51	23.03.2005 10:55:53

Eingangskorb (2) | Stellvertretung | Startbare Vorgänge | **Laufende Vorgänge**

The context menu allows you to access process details.

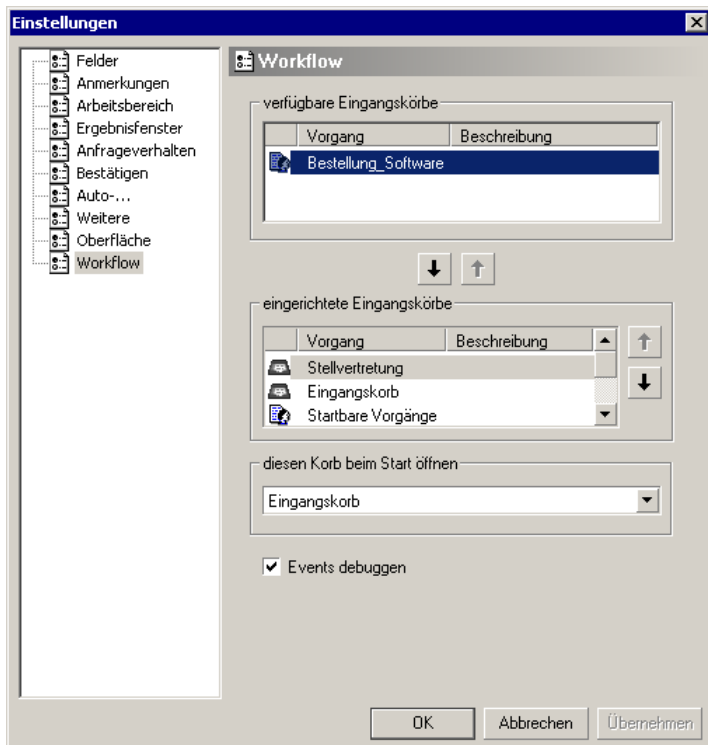
Logging

Activity forms have a second logging tab if the workflow log has been set up. According to the log settings, respective columns are displayed and can be differentiated by shown for each model. The chapter *Process Design* describes all settings available for the integrated process log.



Settings

Users can customize design and behavior of workflow inboxes using the settings dialog. For this purpose, the settings dialog in enaio® client offers a specific area.



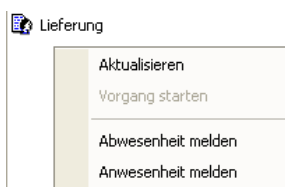
Inbox administration allows users to manage inboxes to be displayed in the workflow area. The following inboxes are active by default:

- | | |
|---------------------|---|
| Inbox | the inbox provides the user with new activities assigned to his roles. |
| Startable processes | process models startable by the user's roles. |
| Substitution | inbox for activities that the user, which is a substitute, receives instead of other users. |
| Running processes | all processes a user currently started are listed here. |

Additionally, the user can set up his own inboxes for process models valid for him in which he collects only processes of the corresponding model.

Substitution

Substitution is related to the users' presence. For example in case of leaves, a user can notify his absence. If all users of a role, for which an activity is designated, are notified as absent, the activity is forwarded to specified substitutes. This does not apply to personalized activities.

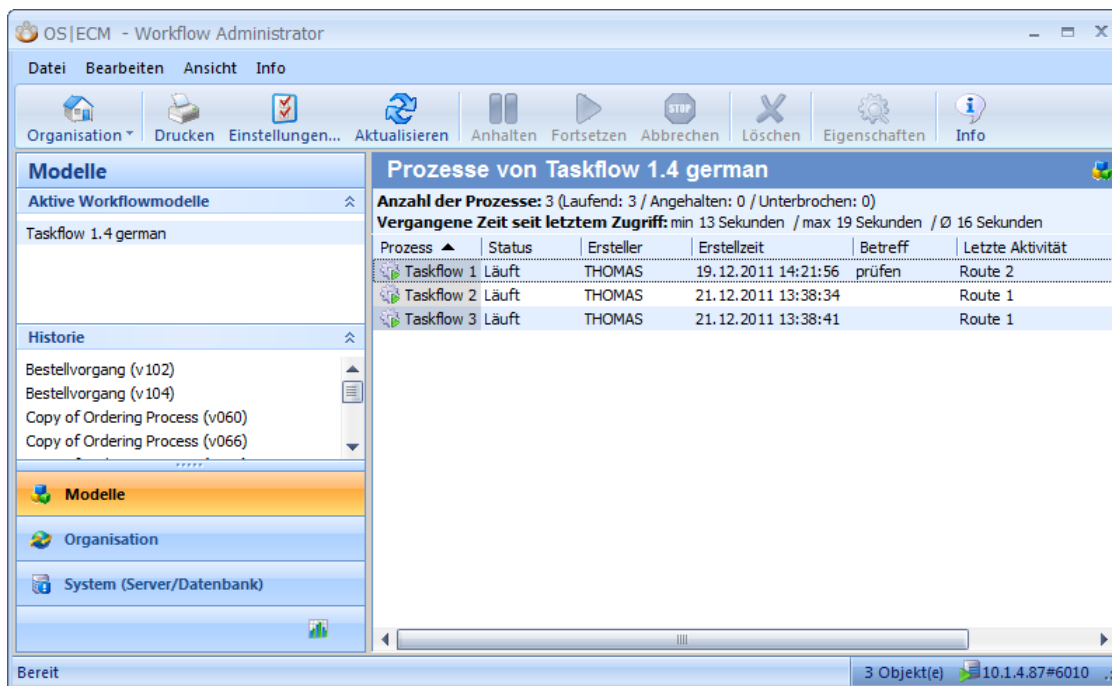


To set the presence status, select the desired entry in the context menu in the workflow area of the enaio® client.

Substitutes are indicated in the organization explorer.

Workflow Administrator

enaio® Workflow Administrator is a tool for monitoring and controlling processes that are already running. It displays all started processes in different groups. The details, in particular the allocation of the workflow variables, can be seen in the process view. Running processes can also be paused, resumed, and canceled. In addition, information on the database and workflow engine status can be called up.



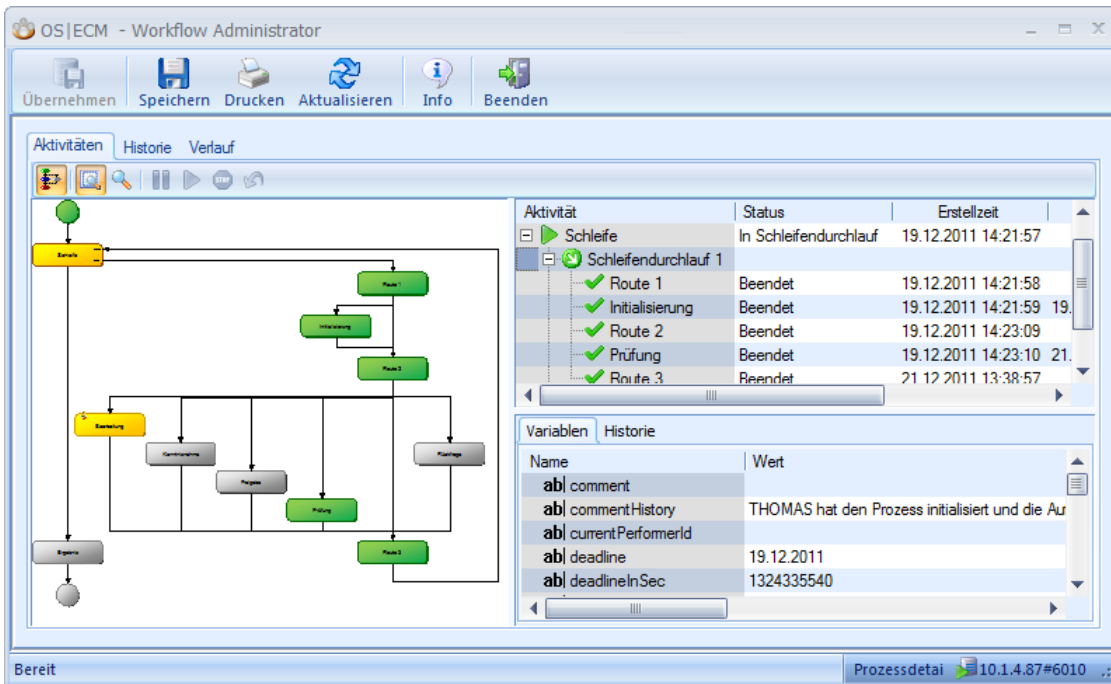
To improve the clarity of the particular areas, different process views can be selected in the Workflow Administrator. Another area allows a view on the database and workflow engine state; the statistics area is used to generate reports.

Models	In the workspace all process models are displayed which are active or in use.
Organization	In the workspace all roles are displayed which are active or in use. If activities are assigned to more than one role or if multiple process activities are active, the assignments to all relating roles are displayed.
Roles	
Organization	Users in roles with pending activities are displayed in the workspace. The list window displays all of the users' activities and associated processes.
User	
System	Information on locks on tables and processes as well as the server status (worker queue) can be viewed.
(server/database)	
Statistics	Configurations can be created with which overview reports or detailed reports can be created.

The following data are displayed in the model view.

Column	Meaning
Status icon	The icon indicates the status of the currently running process.
Process	Name of the process definition
Status	Process status
Creator	User who started the process
Creation time	Time of process creation
Subject	Subject line specified for the model
Last activity	Last performed process activity
Point of time	Time when last activity was finished

Use the context menu of a process to stop, resume or cancel it. Further properties can be viewed in the detailed view. The detailed view is opened through the **PROPERTY** button, the context menu, the function key F2 or a double-click on the process:



This detailed view provides many different views on a single process and offers ways to stop and continue the process and modify variables.

All models with their automatically assigned version number are displayed in the history view of the model area. If a model is modified, the version number increments. The history view contains already finished processes. Except for the column **LAST ACTIVITY** the same columns as in the model view are displayed.

If activities are grouped by roles or users, activities and their respective processes are listed. In such case it may happen that more than one activity is displayed for one running process, in fact if more than one parallel threads are designed in the process.

The following data are displayed in the role and user view.

Column	Meaning
Status icon	The icon indicates the status of the currently running process.
Process	Process model name
Status	Process status
Creator	Name of the user who created the process
Creation time	Time of process creation
Activity	Activity name
Activity status	Activity status
Point of time	Time at which the activity was created
Personalized	Personalization time
By	In role view the name of the user who personalized the activity.
Dunning period	Display of a preset dunning period for the activity

Process and Activity Status

Processes can have different status values which are listed in the following table:

Icon	Process Status
	Initialization
	Running
	Running, at least one activity is stopped
	Stopped
	Active
	Canceled
	Finished
	Interrupted

Activities can have different status values which are listed in the following table:

Icon	Activity status
	Created
	Started
	Executing script
	Checking loop condition
	Running loop
	Activity Created
	Activity Personalized
	Waiting due to a retention period
	Waiting due to a sub-flow
	Executed
	Forwarded
	Completed
	Stopped
	Finished
	Error

Process Properties

Depending on the selected view of the workspace, the assigned processes are listed. For each process, the property dialog can be opened from the context menu.

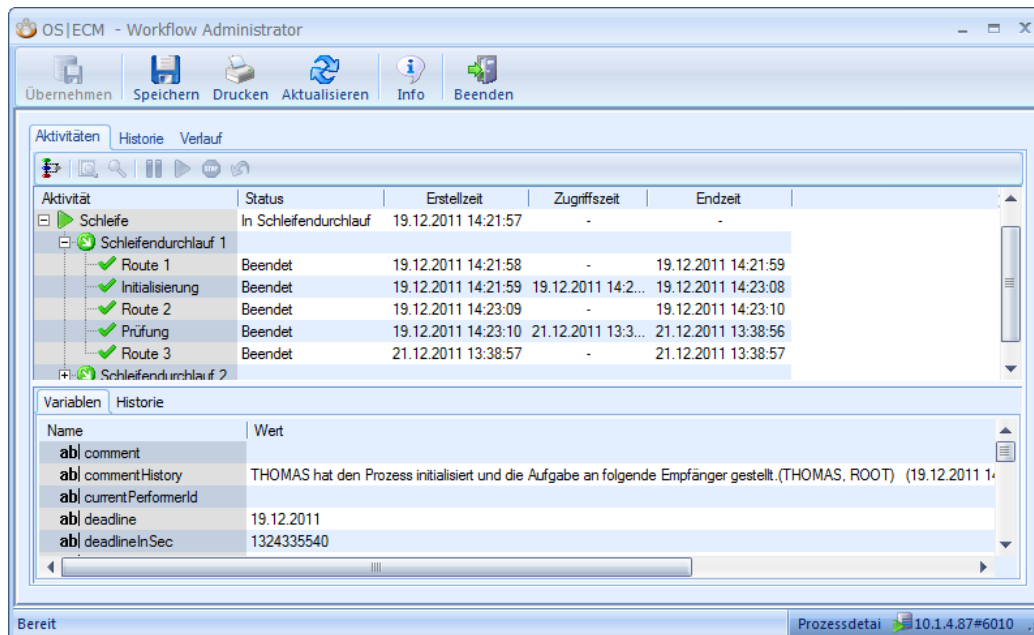
The data are displayed in three tabs:

- **Activities**

All passed activities and the current activity are displayed. The values of the variable of a selected activity are displayed and can be edited.

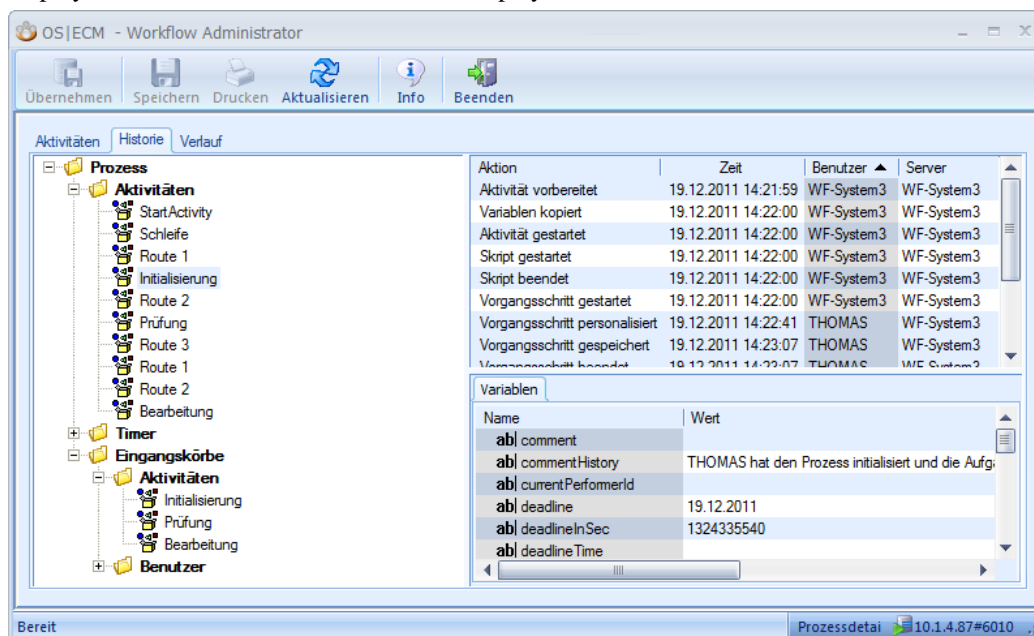
The activity can be assigned to a user or a role with its context menu. If the activity has been personalized, the personalization is removed. Furthermore, activities can be stopped and resumed. Stopped processes can be reset for a particular activity.

This tab is not displayed for finished processes in the history view.



HISTORY

All detailed information concerning the activities and the variables value are displayed on the tab. It can also be displayed in which inboxes activities were displayed.



HISTORY

A history log is displayed on this tab. The log can be saved or printed.

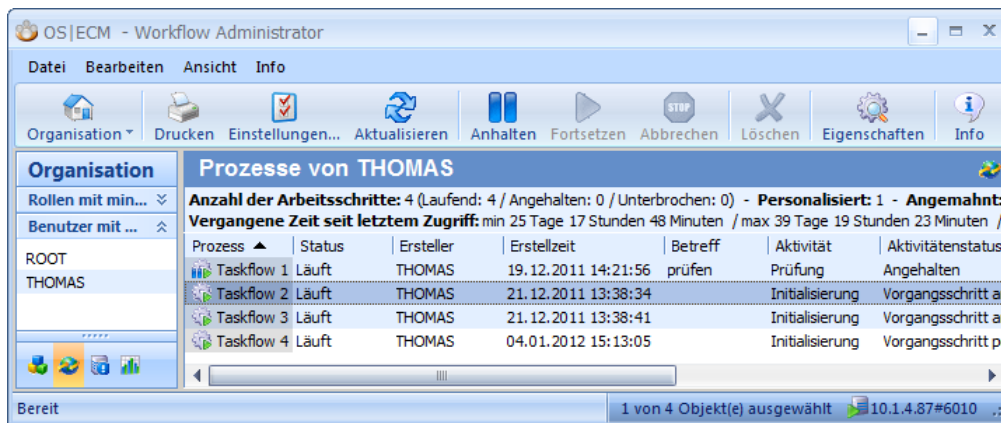
On the **Activities** tab it is possible for stopped processes to reset a process of an activity. All succeeding activities are deleted and variable values are reset. Running periods or the workflow file are disregarded.

Depending on the status that the activity has reached, up to three possible attachment spots are available:

Attachment spot	Explanation
Create activity	The activity is reset to the point of time before the variable transfer.
Execute EndActivity event	The variable is reset to the point of time before executing the EndActivity event.
Forward activity	The activity is reset to the point of time when its execution ended.

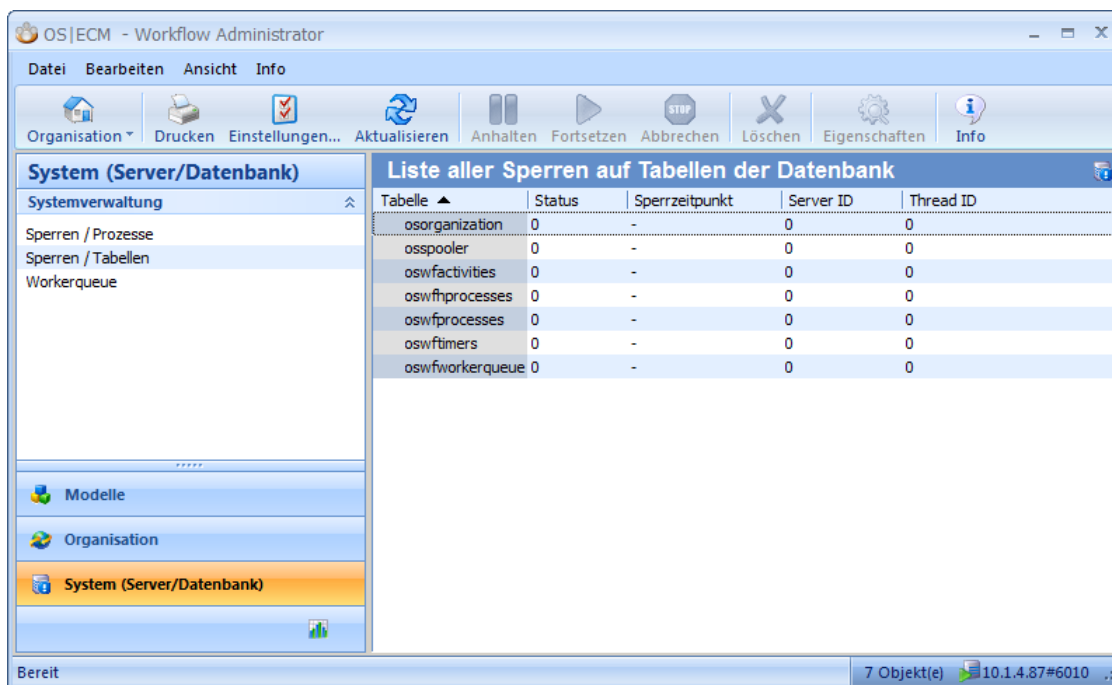
Organization

The **Organization** area provides an overview of running process steps subdivided into roles and users. From the context menu of an entry, it is possible to stop/resume a process step and assign process steps to users or roles. Depending on the activity and the status, these actions can be done for multiple selected process steps at a time.



Server and Database Properties

The **SERVER/DATABASE** area provides an overview of locks on tables and processes as well as the current state of the worker queue (server internal mechanism for process editing) of the workflow engine. Locks can be removed through the context menu.



Statistics Reports

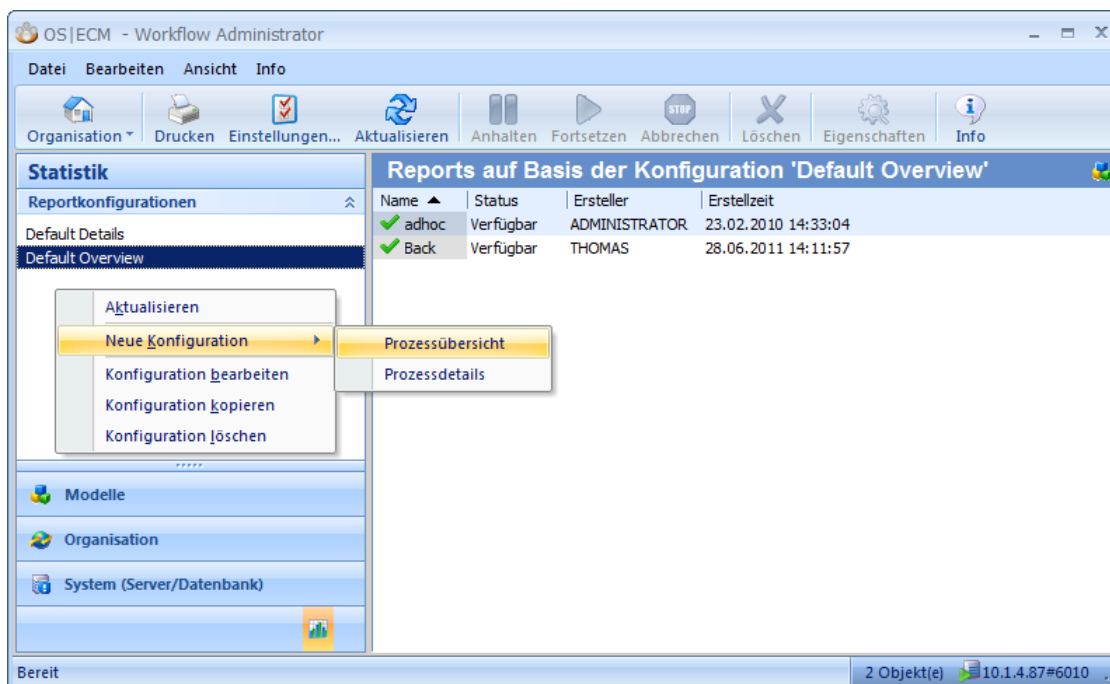
Create report configurations in the **STATISTICS** area. Depending on the configuration you can document with logs the process state in detail or an overview of the state of all processes at a specific point of time.

Based on a detailed or overview configuration a report can be created and viewed in the **Statistics** area or in enaio® client. Configurations can only be created in the **Statistics** area, reports can only be created in enaio® client by users which are authorized in the properties dialog of a workflow family.

Reports are created in the form of XML-files. The files are formatted in enaio® client and in the **Statistics** area or displayed as source code. The structure of these XML files is documented in the appendix (see Report XML). This documentation enables you to provide data for external analysis tools.

Log Configurations

Reports can only be created based on a report configuration. Create report configurations in the **STATISTICS** area.



You can create overview or detailed configurations. A detailed configuration enables you to create reports for an inside view on all activities of a process, an overview configuration enables you to create reports to gain an overview of many running processes.

Process overview configuration

Enter a name for the overview configuration and specify if the processes of all **FAMILIES** or only single families are included, plus enter the degree of detail: overview, itemized list per family or itemized list per family and activity.

You can switch on/off (true/false) recording of single files in great detail.

Allgemein	
Name	
Typ	Prozessübersicht
Ersteller	THOMAS
Erstellzeitpunkt	21/12/2011
ID	B4A98A32E87B44B6853762D434FB5B49
Familien	Keine
Detailltiefe	0 (Übersicht)
Prozess	
Gesamtanzahl der Prozesse	Ja
Anzahl fehlerhafter Prozesse	Ja
Anzahl pausierter Prozesse	Ja
Laufzeit	Ja; Ja; Ja
Aktivitäten	
Gesamtanzahl der Aktivitäten	Ja
Anzahl beendeter Aktivitäten	Ja
Anzahl nicht beendeter Aktivitäten	Ja
Anzahl personalisierter Aktivitäten	Ja
Anzahl fehlerhafter Aktivitäten	Ja
Anzahl pausierter Aktivitäten	Ja
Laufzeit	Ja; Ja; Ja
Laufzeit beendeter Aktivitäten	Ja; Ja; Ja
Laufzeit nicht beendeter Aktivitäten	Ja; Ja; Ja
Laufzeit personalisierter Aktivitäten	Ja; Ja; Ja
Eskalationen (abgelaufene Fristen)	
Anzahl für Prozess	Ja
Gesamtanzahl	Ja

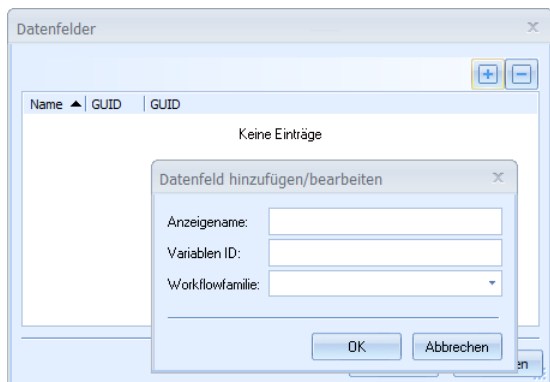
Buttons: OK, Abbrechen

Process details configuration

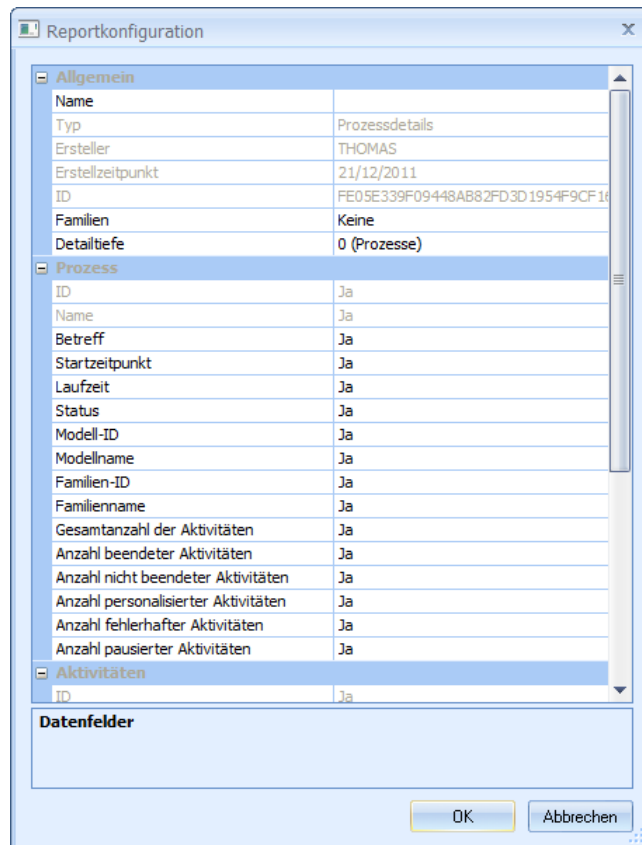
Enter a name for the detailed configuration and specify if the processes of all **FAMILIES** or only single families are included, plus enter the degree of detail: processes or processes and activities.

You can switch on/off (true/false) recording of single files in great detail.

Enter those variables in the **DATA FIELDS** area of which the value is supposed be output.



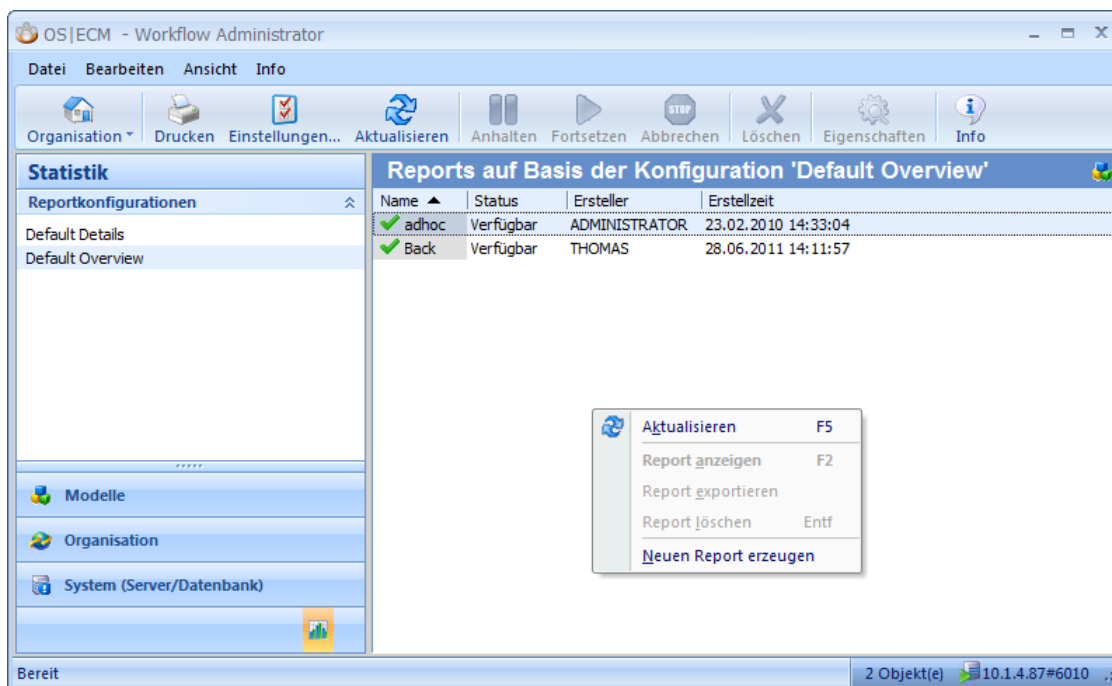
The variable ID can be found in the model overview in enaio® editor-for-workflow.



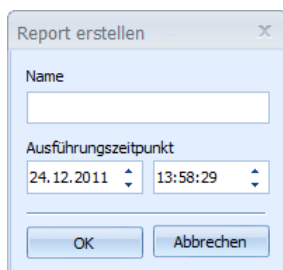
Generate Reports

Reports can only be created based on a report configuration. They can be created in enaio® client by users who have the relevant rights or in enaio® administrator-for-workflow.

Select the designated report configuration and choose from the context menu in the report area **CREATE NEW REPORT**.



Enter a **NAME** and the **EXECUTION TIME** for the report.



The periodic job 'Workflow SpoolerJob' specifies how often existing report requirements are looked for. You can configure this job in enaio® enterprise-manager. An interval of one minute is preset by default.

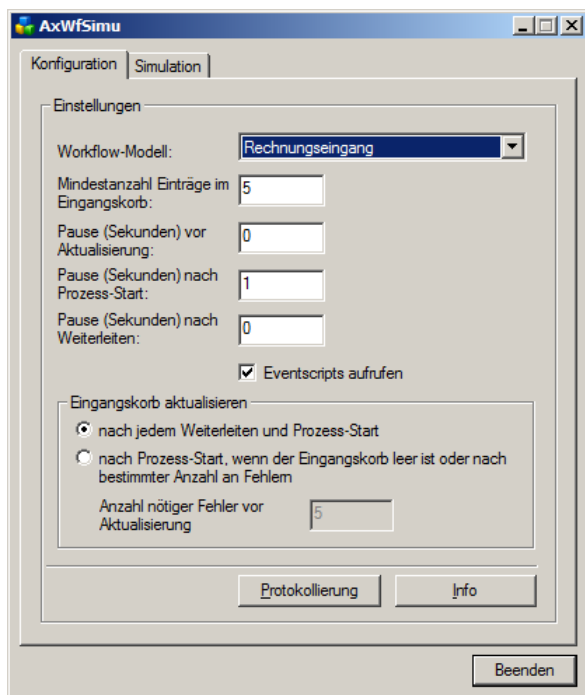
Workflow Simulation

The application `axwfsimu.exe` allows you to automatically test workflow processes. The file is located in the installation directory `...\\clients\\admin`. In the name of the logged in user, processes of a selected workflow model are started and their process steps forwarded. If required, specific event scripts simulate the user's action on forms during the process flow. Run the program `axwfsimu.exe` multiple times simultaneously to simulate multiple workflow participants.

Thus executed processes are recognized as normal workflow processes by the system and, thus can be monitored and statistically analyzed, for example, with the Workflow Administrator.

To launch the application, run the file `axwfsimu.exe`. enaio® client starts automatically. After you have logged in to enaio® client, the configuration dialog opens.

Configuration



The following settings can be edited on the **CONFIGURATION** tab of the file `axwfsimu.exe`:

Choose the model to be simulated from the list of startable processes available for the currently logged in user in the entry **WORKFLOW MODEL**. Only process steps of the selected model are processed automatically during simulation.

With the entry **MINIMUM NUMBER OF INBOX ENTRIES** you can indicate the number of entries to be contained in the user's inbox before one of these inbox elements will be processed. If the inbox contains fewer entries than indicated here, a new process is started instead. If the simulation is supposed to not contain any started process, set this value to 0 to exclusively process the inbox.

Define different waiting periods to not overload the server during simulation with nonstop queries. Entered values express the break in seconds:

BREAK BEFORE UPDATE

A break can be forced during simulation before a server query updates the inbox.

BREAK AFTER PROCESS START

A break can be forced during simulation after a new process has been started.

BREAK AFTER FORWARDING

A break can be forced during simulation after a process step has been forwarded.

Activate the option **ACTIVATE EVENT SCRIPTS** to specify whether or not to execute the client scripts 'BeforeOpen', 'SimulateMaskEdit', and 'BeforeForward' during step processing.

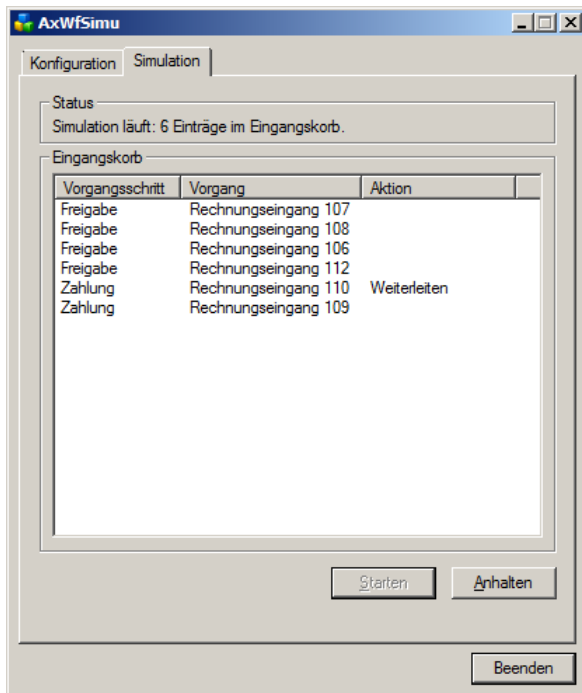
Define the time a server query will update the inbox in the area **UPDATE INBOX**:

Select the inbox update **AFTER FORWARDING AND PROCESS START**. But it will result in higher server load.

When selecting **AFTER PROCESS START IF INBOX IS EMPTY OR AFTER INDICATED NUMBER OF ERRORS**, server load will reduce as less updates are performed. However, with this option you have to accept that the inbox may sometimes not be up to date. For example, process steps that have already been processed by other participants may still be visible in your inbox. When opening such a process step, an error will occur. That is why you can indicate the number of such errors after which the inbox will be updated.

All changed settings will be saved automatically when closing the program `axwfsimu.exe` so they will be available at restart.

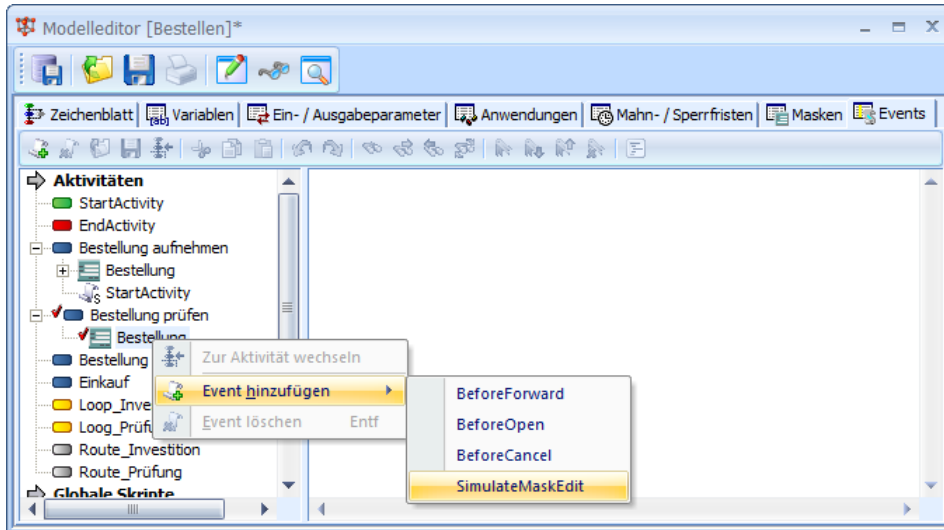
Simulation



Start or stop a simulation in the **Simulation** tab. It is possible to switch between both tabs **Simulation** and **Configuration**, but you cannot change the configuration as long as simulation is running.

During simulation the inbox displays all process steps of the configured model the user can see and which action is currently simulated.

SimulateMaskEdit – Scripts



Since the simulation is performed fully automatically, workflow forms cannot be opened. Nevertheless, user interaction may be simulated. For this purpose, the 'SimulateMaskEdit' event was created and can be set up with the workflow editor. For example, variable values of the process can be manipulated.

The event 'SimulateMaskEdit' is executed after 'BeforeOpen' and before 'BeforeForward' if they all have been activated in the configuration.

Command Line Parameters

The following command line parameters are additionally available for the application `axwfsimu.exe`:

Name	Parameter/Explanation
-wf	Sets the 32-digit ID of the workflow model to be simulated.
-me	Sets the minimum number of entries in the inbox.
-es	If set to 1, event scripts are executed, if set to 0, they are not executed.
-ps	Sets the break in seconds after process start.
-pf	Sets the break in seconds after forwarding a process step.
-pr	Sets the break in seconds before the inbox is updated.
-ra	If set to 1, the inbox is updated after a process step has been forwarded or after process start. If set to 0, the inbox is updated after process start, when there are no entries, or after an indicated number of occurred errors.
-rf	Sets the number of occurred errors after which the inbox is updated.
-up	Sets the user and password automatically to avoid the login dialog, e.g. <code>user@password</code> .
-ao	If set to 1, the simulation starts instantly.

Example:

```
axwfsimu -up user1@pw -es 0 -me 5 -ao 1 -wf 350DD718818F4482B9E708BC6088B61B
```

The program `axwfsimu.exe` is started for 'user1' with password 'pw'. Event scripts will not be executed and the minimum number of inbox elements is five. Processes of the model with the ID '350DD718818F4482B9E708BC6088B61B' will be performed. Simulation starts immediately when launching `axwfsimu.exe`.

Launch `axwfsimu.exe` with the parameter `ao 1` to instantly start the simulation without displaying the `AXWFSIMU` dialog.

Automated Starting of Workflow Processes

Besides starting workflow processes in enaio® client it is possible to initialize workflow processes to transfer documents subsequently to a high-volume indexing or an import.

Furthermore, processes can be started via script code which runs in a workflow start component, but without document transfer.

It is also possible to start workflow processes including e-mail data transfer in Microsoft Outlook.

Integration Using the Import Wizard

enaio® capture is a high-volume indexing component of the enaio® product family. It is used to capture documents digitally in batches, to perform automated or manual indexing, and to consequently import the indexed documents into the document management system. The captured data and documents are assigned to objects in enaio® by the import wizard. This can forward the data and documents to workflow processes for configuring an automated import action or for enaio® capture.

A typical example for the use of enaio® capture is an incoming mail scenario. Documents (e.g. delivery notes, orders, or other receipts) are captured and after gathering the indexing data corresponding editing processes are started. Depending on the type of records, it can be an order, delivery, or other kind of process which requires the participation of multiple roles from different departments. Usually, the records are indexed by the sender and the company reference number is entered, and subsequently forwarded to the respective person in charge or the responsible department. Here, the actual editing of the business process which was designed in the corresponding model takes place.

Workflow Script Component

The workflow-script-component is a command line tool used to execute script code. The script code receives already initialized COM objects (`wfclient` and `assystem`) and can be used to start processes in the script. The workflow-start-component is installed in the `Admin` directory and is called `axwfscript.exe`.

The program can be launched with different command line parameters which are explained below:

Parameter	Type	Meaning	Comment
-u	(optional)	username@password	If user name and password are specified, the login dialog is skipped. Otherwise a normal login (automatic/dialog) is performed.
-s	Required	Script	File name of the script code which will be executed.
-h, -?, /h /?, no parameter		Help	Help text stating possible arguments
-q	(optional)	Quit output	If switched off, console output is suppressed.
-t + seconds	(optional)	Interval	If set, a timer is activated that executes the script in defined intervals. If not set, the program is terminated after script execution.
-srv	(optional)	Server#Port	If specified, the server configuration in the <code>asinit.cfg</code> is ignored and the server specified by parameter is addressed

The -t parameter switches the application to continuous mode and allows scripts to be executed periodically. This option enables realization of the following scenario:

A web server receives requests (orders, invoices) over the internet. The corresponding data is saved in files. This directory is monitored in the script code. When a file is found, the information is read and handed over to the newly started process as input parameters. This allows an automatic connection of sub-systems to the workflow system and at the same time retains flexibility to adjust the data using VB script.

Example

The following command line executes the script c:\test.vbs every 30 seconds under the account of user root. There is no output to the console.

```
Axwfsript. Exe -u root@masterkey -s c:\test.vbs -q -t 30
```

Example Code for starting a Process

The following code can be used to start processes. It verifies in the collection of startable processes if a process model with the name 'Order' exists and - if this is the case - returns the start parameters of the process preset with values.

```
'Get startable processes
Set wfitems = wfclient.wfitems
For i =0 To wfitems.count -1
    Set wfitem = wfitems.item(i)
    If wfitem.name = "Order" Then
'Set start variable
        Set inputvars = wfitem.inputvariables
        inputvars.Item("Variable1").value = "abx"
        inputvars.item("Variable2").value = "12434"
'Start process
        iRet = wfitem.startprocess (Nothing, inputvars, varid, varerror)
        If iRet <> 0 Then
            msgbox "Error in StartProcess " & iRet
        Else
            msgbox "Process started"
        End If
    End If
Next
```

The workflow script program is licensed for each workstation with the license key AVG.

Microsoft Outlook Add-In

Workflow processes can be started with Microsoft Outlook during the distribution of e-mail data. This function is made available by the COM-add-in oxvboutl.dll. During installation, the oxvboutl.dll add-in is copied to the clients\client32 directory and registered.



Workflow A **WORKFLOW** button will be integrated into the toolbar of Microsoft Outlook. This button will open an assignment dialog.

The user selects a startable process and populates the interface variables with e-mail data or any free text. Interface variables are variables set up as input variables for enaio® editor-for-workflow.

The configuration is carried out in the configuration file oxoutl_9.cfg in the clients\client32 directory.

Without a configuration the user can select his startable process and preset the interface variable with any e-mail data or free-text. The e-mail file or attachments cannot be transferred without a configuration to the workflow file.

The configurations can specify for each user which startable processes will be available with the Outlook add-in. The assignment of e-mail data for interface variables can be preset or pre-configured. E-mail file or attachments can be transferred to the file.

The configuration file oxoutl_9.cfg can be edited with any editor. It is used to configure automated indexing while transferring e-mails into the DMS (see Administration Handbook).

A section is set up for each workflow process which will be configured. The section contains configurations for interface variables and transfer of e-mail files and e-mail file attachments.

Variable Configuration

A section is set up for each workflow process which will be configured:

```
[WFPROCESS_ 'ProcessName']
```

Example: [WFPROCESS_MailInbox]

The assignment dialog with which a user starts processes, lists all interface variables automatically. If variables are supposed to be hidden, the process section must have the following entry:

```
WFVARIABLESHOW_ 'VariableName'=false
```

Thus hidden variables are not visible for users, but can be preset with values with a configuration entry. Visible variables can be preset with values with a configuration entry and made write-protected, in order to avoid user changes of variable presets.

Variables which are not listed in the process section can be preset by the user with content of an Outlook field or with free-text.

Variables which have an Outlook field assigned can be preset with content of another Outlook field, but not with free-text.

Variables to which a configuration assigns text can be preset with any other free-text or the content of an Outlook field.

Variables with a list assigned can be preset by the user with a list entry, but neither with free-text nor with the content of an assigned Outlook field.

Outlook field

The assignment of an interface variable to an Outlook field is achieved with the following entry:

```
WFVARIABLECONNECTED_ 'VariableName'=Outlookfield
```

The following Outlook fields can be used:

- From
- To
- CC
- BCC
- Date
- Subject
- Message

Example: WFVARIABLECONNECTED_Editor=To. The interface variable 'Editor' is preset with the content of the Outlook field 'To', the recipient of the e-mail.

The content of the Outlook field 'Date' is converted into the following date format: dd.mm.yyyy. The data format can also be specified.

Example: DateFormat="dd.mm.yyyy hh:mm".

Text

The assignment of text to an Outlook field is achieved with the following entry:

```
WFVARIABLECONNECTED_ 'VariableName'=# 'text' #
```

The text will be bracketed with the # character. The text can contain any number of characters. The following variables can be used.:

- #user#-OS-user name
- #date#-Date in the system settings format
- #time#-Time in the system settings format
- #fulldate# - Date in the defined format (see above)

Example: WFVARIABLECONNECTED_Editor=#Doctor #user#. The interface variable 'Editor' is preset with the text 'Doctor', a space and the name of the OS-user.

The OS-user name does not need to be identical with the person description in the workflow organization.

List

The assignment of an interface variable to list is achieved with the following entry:

```
WFVARIABLECONNECTED_ 'VariableName'=#List#ListSectionName
```

A section with the corresponding name is created for the list entries. The list can be referenced from any process section.

Example:

```
[List Section Name]
DEFAULT=Order
value1=Offer
value2=Order
value3=Order confirmation
value4=Delivery note
value5=Invoice
```

The list entries are numbered sequentially. A list entry can be set as a default preset. This entry is then preset.

Write protection

When the assignment is write-protected, the user cannot change the assignment of an Outlook field or a text for interface variables.

An assignment is write-protected with the following entry:

```
WFVARIABLECHANGE_ 'VariableName' = false
```

Data Transfer Configuration

The e-mail file and the e-mail attachments can be transferred to the workflow file.

All file types which are supposed to be transferred are listed in the process section.

```
FILEFILTER=#msg#pdf#bmp#
```

Independently of the e-mail format (msg, mime) 'msg' must be specified as 'Filefilter' parameter for the e-mail transfer. Independently, the format with which the e-mail is transferred, is specified in the section 'General' or in the entry 'Format' in the process section (see below).

Each file type can have cabinet and document type assignments specified. In addition, the main type of the document type must be specified. The main type defines with which component the file is displayed.

```
FILEMAINTYPE_ 'FileType' = 'MainType'
FILEARCHIVENAME_ 'FileType' = 'CabinetName'
FILENAME_ 'FileType' = 'DocumentTypeName'
```

Example:

```
FILEMAINTYPE_pdf=4
FILEARCHIVENAME_pdf=Addresses
FILENAME_pdf=Letter
```

Attached PDF-files are assigned to the cabinet 'Addresses' and the document type 'Letter'. The PDF-file is a Windows document type. It is always opened with the application that in enaio® administrator has been set up for the document type 'Letter'.

The following DMS types can be set up:

Main type	Document type	File formats
1	Grayscale Images	JPEG
2	Black/White images	TIFF G4
3	Color images	JPEG
4	Windows	Application format
5	Moving images	MPEG, AVI
6	E-mail	MIME, MSG
7	XML	XML, XSL, XSLT

E-mail files can be transferred in 'MIME' or 'MSG' format. The format can be specified or selected by the user. The default setting is 'MIME'. If the format is specified in the 'General' section, the entry is valid for all process sections in which no format has been specified. '1' stands for 'MIME', '2' for 'MSG'. Set '3' to allow the user to select the format.

Example: Format=3

If file transfer is not configured, the user can select if he wants to transfer a file or not. The entry

`FILECHANGE_ 'FileType' = false` in the process section enables you to define in the configuration that files must be transferred.

DMS-ID and the object type name of an interface variable can be assigned to document types with an assignment to a DMS location. If more than one file is transferred to the workflow file, the files are listed separated by a comma.

`FILEID_ 'FileType' = 'VariableName'`

`FILETYPE_ 'FileType' = 'VariableName'`

User Assignment Configuration

The user can start all workflow processes which are startable from the register 'Startable Processes' with the Outlook add-in, even if no interface variable or no file transfer were defined for the processes.

Workflow processes can be hidden for all users with entries in the [General] section. A [USER_ 'OS-UserName'] section can be set up for each user in order to show/hide workflow processes for this user.

The OS-user name does not need to be identical with the person description in the workflow organization.

Example:

```
[General]
```

```
WFPROCESS_Order=false
```

```
WFPROCESS_Purchase=false
```

```
[USER_Peterson]
```

```
WFPROCESS_Purchase=true
```

```
WFPROCESS_Requirement=false
```

The workflow processes 'Order' and 'Purchase' are hidden for all users. User 'Peterson' can start the 'Purchase' process as this process is shown in his section. He cannot start the 'Order' process though, as this process is hidden for him.

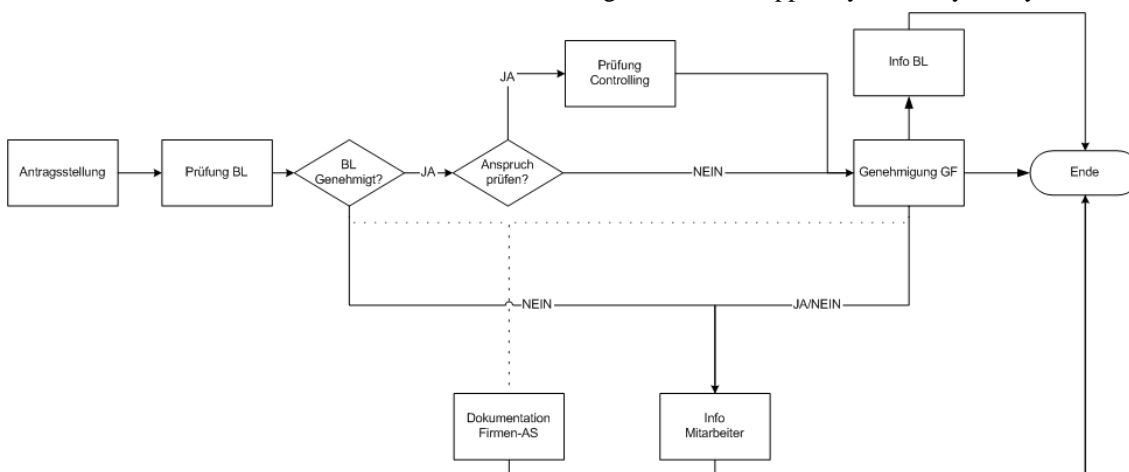
Workflow Examples

Leave Workflow

The process starts with the employee filing his application. Initially, the leave entitlement was entered in a designated database for all employees at the company. While filing his application the employee chooses between leave and compensatory time off. The difference between the two modes is that compensatory time off is not counted as leave entitlement. The employee selects the start and the end of the leave period. It must be possible to request half days. Upon forwarding the remaining leave entitlement is automatically calculated.

Furthermore the employee can request his leave entitlement to be verified by the controlling department. To do so he has to mark the 'Check leave entitlement' checkbox.

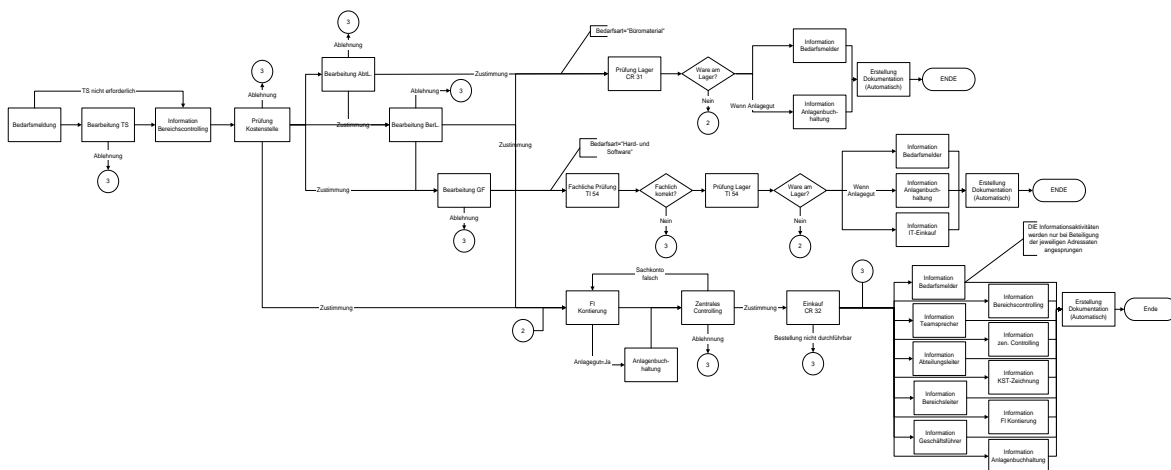
The application is then forwarded to the responsible department manager. He verifies the application and decides if the employee can go on holiday at the specific time. The responsible department manager is determined with the organizational structure. If this is impossible due to functional reasons, the section to which the employee belongs must be entered in the leave database. The selection of the target role must happen dynamically in any case.



For compensatory time off applications, the manager verifies the claim for compensatory time off. The manager approves the application by marking the checkbox. Upon approval it is verified if the employee has requested his leave entitlement to be verified. If this is the case, the application is forwarded to the controlling department. If no verification was requested, the application is forwarded to the chief executive officer. Upon rejection by the department manager the employee is notified (activity info employee) and the process is documented in the DMS. Thus, the application is processed by the chief executive officer. Upon approval the employee and manager are notified, the leave is documented in the DMS and in the general Outlook calendar of the company. Upon rejection the same notifications take place and everything is documented in the DMS.

Order Workflow

This example illustrates a more complex processing of a purchase requisition with additional verification of competence guidelines depending on cost center, employee position and order value.



The single process steps are explained in detail in the following section.

Purchase Requisition

A purchase requisition can be performed by any employee of the company. The process can be started once the statement of work has been defined. This can be done informally in writing and is added to the process. Potential offers which have already been requested by the person who informed about the demand can be added to the purchase requisition. (*Definition: the person who informs about the demand is the employee who starts the purchase requisition process.*). The person informing about the demand specifies the cost center. Furthermore he defines the demand (*Definition: the demand specifies the order content. Depending on the demand the corresponding competence guideline is chosen.*). The workflow depicts all demand types of a competence guideline.

Documentation in DMS: It is specified that the documentation of purchase requisition processes in the DMS takes place in its own cabinet (logical archive). An analogous data model is part of the system. The described cabinet in enaio® is defined as the "order archive."

Roles: purchase reporter

Approval according to Competence Regulation

Upon approval by the individual hierarchy stages, the process is forwarded according to the competence regulation. Upon approval by the hierarchy stage TS (team spokesperson), the divisional controlling department participates informally. At this stage the required budget will be verified. This information is depicted in the workflow.

Subsequently the process is passed on to the next approval according to the competence regulation. For requirement types which do not require the team spokesperson and department manager to participate in the approval process, the divisional controlling department is directly integrated after the purchase requisition.

If the purchase requisition gets rejected in the competence line, the purchase reporter and the section controlling has to be informed. All information and documents resulting from this process is filed in the purchase order archive. The process will then be stopped.

Roles:

1. Divisional controller
2. Team spokesperson
3. Department manager
4. Chief executive officer

Special Rules according to Requirement Type

For the requirement types 'Purchases to be warehoused' and 'Office and business equipment' it is checked whether these goods are available in stock. The warehouseman checks available goods.

If the goods are in stock, a feedback is sent to the purchase reporter informing him when and where he can pick up the goods. If the ordered goods are assets (order value > € 410), the asset accounting will be informed at FI. This is done with the 'Change feedback for assets' form. The form is filled in by the warehouseman.

If the goods are not in stock, this information is documented in the workflow. No further actions need to be taken by the warehouseman.

Subsequently a forward to the FI role takes place.

- The requirement type 'Hardware and software' is checked by a professional (role: professional IT purchasing agent). He checks whether or not ordered hardware and software suit the company's system concept. A NO has to

be documented in the process. ALL process participants will be notified. The process will then be stopped. The purchase process is not supposed to be restarted by the purchase reporter changing the order. The purchase reporter must start a new purchase requisition process. He can reuse documents in the purchase order archive and, for example, does not need to again write the statement of work.

If YES (order is correct), it is checked whether the goods are in stock. If the goods are in stock, a feedback is sent to the purchase reporter informing him when and where he can pick up the goods. If the ordered hardware and software goods are assets (order value > € 410), the asset accounting will be informed. This is done with the 'Change feedback for assets' form. The form is filled in by the professional IT purchasing agent. In addition, an employee with the role 'IT purchasing department' must be notified. If the goods are not in stock, this information is documented in the workflow. No further actions need to be taken by the professional IT purchasing agent. Subsequently a forward to the FI role takes place.

Roles:

1. Warehouseman
2. Asset accountant
3. Professional IT purchasing agent
4. IT Purchasing department

FI-Financial Accounting

In this process step the person in charge FI enters a ledger account into the purchase requisition. In case it is an asset, the asset category is indicated in the purchase requisition and an asset master record created in the SAP.

The person in charge documents his proper workflow actions by ticking off the check box *Allocation done*.

Roles:

Allocation (FI)

Controlling

This process step includes a check carried out by the central controlling department. A check box in the workflow documents approvals and rejections. The controlling department has to explain and document its rejection in an annotation. In such a case, ALL participants will be notified. The process will then be stopped. If the controlling department gives its approval, the process step will be forwarded to the purchasing department.

Roles:

Central controlling department

Purchase

The purchasing department determines suppliers. This information is neither documented in nor supported by the workflow. Requests for quotation are submitted to selected suppliers in SAP. These are filed in the purchase order archive by SAP ArchiveLink. SAP allocates a number for each request for quotation. These numbers are documented in the workflow/purchase order archive.

In the following, quotations are awaited to be sent. Each quotation is at first sent to the purchase reporter for revision. If found correct as regards the contents, the quotation is sent to the purchasing department for commercial checking. Quotation checks must be supported by the workflow.

The purchasing department decides at set date for one of the received quotations and places the order.

Roles:

Purchase<requirement type> (e.g. purchase hard- and software)

Purchase Order

The purchase order is generated in SAP. The then generated purchase order number must be documented in the workflow. A notification on the purchase reporter including the purchase order number and the expected date of delivery is created. The purchase order is filed in the purchase order archive by SAP ArchiveLink.

Roles:

Purchase<requirement type> (e.g. purchase hard- and software)

Process Completion

A log containing all relevant process data is automatically created and filed in the purchase order archive.

- Combined structured and ad hoc workflow for flexible process editing for approval procedures

- The incoming mail scenario (centralized capturing of documents at the mail administration center and automatic forwarding to the corresponding departments and persons in charge, including approval of response letters and filing in the DMS)
- Invoice receipt (centralized capturing of invoices at the mail administration center, approval with regards to the content, administrative and financial clearance by different departments, including transfer to the accounting department)

Appendix

Analysis Sheet for Business Processes

Verbalized Description of the Business Process

Business Process Participants

Activity Summary

No.	Activity	Description	Participating roles
1	Start activity	<this activity is available in all models>	<here the roles are entered that can start the process>

Input Forms

Activity 1:

Activity n:

Transitions between Activities

No.	From activity	After activity	Condition

Activities to be executed simultaneously

No.	Activity	Description	Participating roles

Workflow File

For which activities is a workflow file available?

No.	Activity	Available document types	Minimum Number	Maximum Number

Can processes be started in OS|CLIENT with objects?

What will happen with the documents of the workflow file at the end of a process?

Client Scripts

Which events (initializations, plausibility checks, etc) will be executed while editing activities?

No.	Activity	Event type	Description

Programming Reference

Environment

The programming language Microsoft Visual Basic Script is used to program event code and other scripts in the workflow engine. This programming language is available on all Windows systems and provides extensive possibilities to access the operating system environment, the file system, databases and objects. A detailed description of the programming language can be accessed in the MSDN of Microsoft development environments or the Internet under <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/script56/html/vbscripttoc.asp>.

This section briefly describes the language's most important aspects. The script language offers some simple functions. More functions can be accessed by referencing to operating system internal objects, e.g. the File System Object or the Active Data Object, as well as to additionally set up scriptable objects.

Information about events in enaio® webclient can be found in the 'Workflow Event-Code for OSWEB' handbook. The document is located in the directory 'Win32\Disk1\OSWEB\Doc' on the installation DVD.

Standard VB Script Functions

By default the following functions are offered in VB script. They can be accessed directly from the script code without prior object initialization.

Abs	Array	Asc	Atn
CBool	CByte	CCur	CDate
CDbl	Chr	CInt	CLng
Conversions	Cos	CreateObject	CSng
Date	DateAdd	DateDiff	DatePart
DateSerial	DateValue	Day	Derived Maths
Eval	Exp	Filter	FormatCurrency
FormatDateTime	FormatNumber	FormatPercent	GetLocale
GetObject	GetRef	Hex	Hour
InputBox	InStr	InStrRev	Int, Fixs
IsArray	IsDate	IsEmpty	IsNull
IsNumeric	IsObject	Join	LBound
LCase	Left	Len	LoadPicture
Log	LTrim; RTrim; and Trims	Maths	Mid
Minute	Month	MonthName	MsgBox
Now	Oct	Replace	RGB
Right	Rnd	Round	ScriptEngine
ScriptEngineBuildVersion	ScriptEngineMajorVersion	ScriptEngineMinorVersion	Second
SetLocale	Sgn	Sin	Space
Split	Sqr	StrComp	String
Tan	Time	Timer	TimeSerial
TimeValue	TypeName	UBound	UCase
VarType	Weekday	WeekdayName	Year

A detailed description of each function and required parameters can be found at the locations listed above.

Active Data Object

The active data object allows access to any data source with an ODBC or an OLEDB-connection.

```
On Error Resume Next
'Enter Connectstring here
'Example: sConnect="PROVIDER=SQLNCLI;DSN=acm520;UID=sysadm;PWD=friend;"
Set db = CreateObject("ADODB.Connection")
If db Is Nothing Then
    'no ADO available
Else
With db
    .CursorLocation = adUseClient
    .Open sConnect
    If .State <> 1 Then
        'error in Connect
        msgbox "Error in database connection" & vbCrLf & err.description
    Else
        msgbox "OK"
    End If
End With
End if
```

Subsequently an SQL-statement can be executed using the `Execute` method. In case of an SQL-Select-statement a recordset with data sets is received. A detailed description of the objects and methods for ADO are available in MSDN on the internet <http://msdn.microsoft.com>.

File System Object

The File System Object-Interface is part of the Microsoft Windows operating system. This interface can be used to e.g. copy, rename and delete files and directories. Furthermore, it allows text files to be read and written.

The following example illustrates the access of the File System object.

```
Dim fso, MyFile
Set fso = CreateObject("Scripting.FileSystemObject")
Set MyFile = fso.CreateTextFile("c:\testfile.txt", True)
MyFile.WriteLine("This is a test.")
MyFile.Close
```

A complete reference to the methods of the File System Object-Interface can be found here:

<http://msdn.microsoft.com/library/default.asp?url=/library/en-us/script56/html/sqfilesystemobjects.asp>

Example source code and extensive descriptions may also be found under this address.

Interface Objects

COM interfaces are provided for server and client events. These allow access to workflow models, organizational structures, processes, and activities as well as workflow files and documents. Complete technical documentation on objects and methods including example source code can be found in the 'COM Objects for enaio®-Workflow' document.

Report Configuration XML

General

StatisticReportConfiguration Tag

Each report configuration is enclosed by a StatisticReportConfiguration tag. It has the following attributes:

Id	Report identification
Name	Report name
Type	Report type (process overview = 0, process details = 1)
OrganisationId	Id of the organization for which the report generated
CreatorId	Id of the creator of the configuration
CreatorName	Name of the creator of the configuration
CreationTime	Time at which the configuration was created (time stamp)
Version	Main version of the report (e.g. 550)
SubVersion	Sub version of the report (e.g. 1)
MinProcessVersion	Main version (e.g. 550) that at least had to run at the time of process creation so the process is entered into the statistics.
MinProcessSubVersion	Sub version (e.g. 0) that at least had to run at the time of process creation so the process is entered into the statistics.

Process Overview Report Configuration

Rows Tag, DrillDownType Tag, FamilyIds and FamilyId Tag

The Rows tag is included in the StatisticReportConfiguration tag and has no attributes. It contains the DrillDownType and FamilyIds tags.

The following values are available for the DrillDownType tag:

0	Creates a general survey not resolved by families and activities.
1	Creates a general survey and detailed list for each family.
2	Creates a general survey and detailed list for each family and activity.

FamilyIds group FamilyId tags; these contain the family identifiers of which processes need to be included in reports that are created by this configuration. If FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF (32 x F) is used as the family identifier, all families of the organization will be used.

Columns Tag

The Columns tag is included in the StatisticReportConfiguration tag and has no attributes. The following tags which each have an attribute called Active, that activates/deactivates the respective option (Active = 1), are listed within the Columns tag. The following report options are listed as tags within the Columns tag.

NumberOfProcesses	Total number of processes
NumberOfActivities	Total number of all activities
NumberOfActivitiesUnfinished	Number of not finished activities
NumberOfActivitiesFinished	Number of finished activities
NumberOfActivitiesPersonalized	Number of personalized activities
NumberOfProcessesError	Number of erroneous processes
NumberOfActivitiesError	Number of erroneous activities

NumberOfProcessesPause	Number of paused processes
NumberOfActivitiesPause	Number of paused activities
MinProcessRunTime	Minimum process runtime
MeanProcessRunTime	Average process runtime
MaxProcessRunTime	Maximum process runtime
MinActivityRunTime	Minimum activity runtime
MeanActivityRunTime	Average activity runtime
MaxActivityRunTime	Maximum activity runtime
MinActivityRunTimeFinished	Minimum runtime of finished activities
MeanActivityRunTimeFinished	Average runtime of finished activities
MaxActivityRunTimeFinished	Maximum runtime of finished activities
MinActivityRunTimeUnfinished	Minimum runtime of unfinished activities
MeanActivityRunTimeUnfinished	Average runtime of unfinished activities
MaxActivityRunTimeUnfinished	Maximum runtime of unfinished activities
MinActivityRunTimePersonalized	Minimum runtime of personalized activities
MeanActivityRunTimePersonalized	Average runtime of personalized activities
MaxActivityRunTimePersonalized	Maximum runtime of personalized activities
NumberOfEscalationsProcess	Total number of escalations (expired deadlines)
NumberOfEscalationsActivity	Total number of escalations (expired deadlines) of activities
NumberOfEscalationsActivityUnfinished	Number of escalations (expired deadlines) of unfinished activities
NumberOfEscalationsActivityFinished	Number of escalations (expired deadlines) of finished activities
NumberOfEscalationsActivityPersonalized	Number of escalations (expired deadlines) of personalized activities

Detailed Process Report Configuration

Rows Tag, DrillDownType Tag, FamilyIds and FamilyId Tag

The Rows tag is included in the StatisticReportConfiguration tag and has no attributes. It contains the DrillDownType and FamilyIds tags.

The following values are available for the DrillDownType tag:

0	Process entries are created, but single activities are not listed.
1	Process entries and activities are listed.

FamilyIds group FamilyId tags; these contain the family identifiers of which processes need to be included in reports that are created by this configuration. If FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF (32x F) is used as the family identifier, all families of the organization will be used.

Columns Tag

The Columns tag is included in the StatisticReportConfiguration tag and has no attributes. The following tags which each have an attribute called Active, that activates/deactivates the corresponding option (Active = 1), are listed within the Columns tag. The following report options are listed as tags within the Columns tag.

Process data:

ProcessId	Process identification
ProzessName	Process name

ProcessSubject	Process subject
FamilyId	Family identification
FamilyName	Family name
ModelId	Model identification
ModelName	Model name
ProcessStartTime	Start time of the process
ProcessRunTime	Running time of the process
ProcessState	Process status
NumberOfActivities	Total number of all activities
NumberOfActivitiesUnfinished	Number of not finished activities
NumberOfActivitiesFinished	Number of finished activities
NumberOfActivitiesPersonalized	Number of personalized activities
NumberOfActivitiesError	Number of erroneous activities
NumberOfActivitiesPause	Number of paused activities
NumberOfEscalationsProcess	Number of escalations (expired deadlines) in the process
NumberOfEscalationsActivity	Total number of escalations (expired deadlines) of activities
NumberOfEscalationsActivityUnfinished	Number of escalations (expired deadlines) of unfinished activities
NumberOfEscalationsActivityFinished	Number of escalations (expired deadlines) of finished activities
NumberOfEscalationsActivityPersonalized	Number of escalations (expired deadlines) of personalized activities

Activity data (for DrillDownType = 1)

ActivityId	Activity identification
ModActivityId	Activity identification in the model
ActivityName	Activity name
ActivityStartTime	Start time of the activity
ActivityRunTime	Activity runtime
ActivityState	Activity status
NumberOfEscalationsActivity (the same tag as in process data)	Number of escalations (expired deadlines) in the activity

ProcessDatafields Tag, ProcessDatafield Tag

The ProcessDatafields tag is included in the Columns tag and has no attributes. It can contain multiple ProcessDatafield tags, and identify global process variables from which values are output in the report. The following attributes are available for the ProcessDatafield tag:

Id	Identifier of the process variables
FamilyId	Identifier of the family Id for which the process variable will be output in the report
Name	Display name for the process variable in the given family

Report XML

Reports are processed for the user in HTML format including graphical elements. Additionally, it is possible to save generic reports as XML files in order to analyze statistical data in greater detail. The XML format is explained in the following section.

SR Tag (StatisticReport)

Each report is enclosed by an SR tag. It always contains exactly one StatisticReportConfiguration tag that describes the underlying configuration (see above) as well as exactly one D tag that contains statistically determined data (see below). The following attributes are available for the SR tag:

Id	Report identification
N (Name)	Report name
CrT (CreationTime)	Time stamp of report creation
CrY (CreationYear)	Year of report creation
CrM (CreationMonth)	Month of report creation
CrD (CreationDay)	Day of report creation
CrH (CreationHour)	Hour of report creation
CrU (CreationMinute)	Minute of report creation
CrS (CreationSecond)	Second of report creation
CrI (CreatorId)	Identification of the report creator
CrN (CreatorName)	Name of the report creator

Process Overview Report

D Tag (Data)

The D tag is included in the SR tag and has no attributes. It always contains an OD tag and, according to the 'DrillDownType' of the report configuration, also an AD and/or an FD tag.

OD Tag (OverviewData), FIs Tag (FamilyIds), FI Tag (Family Id)

The OD tag is located inside the D tag and provides summarized information on running processes. Furthermore, it contains FI tags (grouped by an FIs tag) that identify all families if which processes can be viewed in the report. The OD tag has the following attributes (if the corresponding options are activated in the report configuration):

NP (NumberOfProcesses)	Total number of running processes
NPS (NumberOfProcessesPause)	Number of paused processes
NPE (NumberOfProcessError)	Number of erroneous processes
sP (EscalationsProcesses)	Total number of escalations (expired deadlines) of all processes
iPR (MinimumProcessRuntime)	Minimum process runtime
ePR (MeanProcessRunTime)	Average process runtime
aPR (MaximumProcessRunTime)	Maximum process runtime
NA (NumberOfActivities)	Total number of all activities of running processes
NAF (NumberOfActivitiesFinished)	Number of finished activities
NAE (NumberOfActivitiesError)	Number of erroneous activities
NAS (NumberOfActivitiesPause)	Number of paused activities
NAU (NumberOfActivitiesUnfinished)	Number of not finished activities
NAP (NumberOfActivitiesPersonalized)	Number of personalized activities
sA (EscalationsActivities)	Total number of escalations (expired deadlines) of all activities
sF (EscalationsActivitiesFinished)	Total number of escalations (expired deadlines) of finished activities
sU (EscalationsActivitiesUnfinished)	Total number of escalations (expired deadlines) of unfinished activities
sAP (EscalationsActivitiesPersonalized)	Total number of escalations (expired deadlines) of personalized activities
iAR (MinActivityRunTime)	Minimum activity runtime

iAF (MinActivityFinishedRunTime)	Minimum runtime of finished activities
iAU (MinActivityUnfinishedRunTime)	Minimum runtime of unfinished activities
iAP (MinActivityPersonalizedRunTime)	Minimum runtime of personalized activities
eAR (MeanActivityRunTime)	Average activity runtime
eAF (MeanActivityFinishedRunTime)	Average runtime of finished activities
eAU (MeanActivityUnfinishedRunTime)	Average runtime of unfinished activities
eAP (MeanActivityPersonalizedRunTime)	Average runtime of personalized activities
aAR (MaxActivityRunTime)	Maximum activity runtime
aAF (MaxActivityFinishedRunTime)	Maximum runtime of finished activities
aAU (MaxActivityUnfinishedRunTime)	Maximum runtime of unfinished activities
aAP (MaxActivityPersonalizedRunTime)	Maximum runtime of personalized activities

FD Tag (FamilyData), FE Tag (FamilyDataEntry)

The FD tag is included in the D tag, groups all FE tags and has no attributes. The FD tag only exists after being specified in the configuration by the 'DrillDown' type. The FE tag provides information on a process family and has the following attributes (if the corresponding options are activated in the report configuration):

FI (FamilyId)	Family identification
FN (FamilyName)	Family name
NP (NumberOfProcesses)	Total number of running processes
NPS (NumberOfProcessesPause)	Number of paused processes
NPE (NumberOfProcessError)	Number of erroneous processes
sP (EscalationsProcesses)	Total number of escalations (expired deadlines) of all processes
iPR (MinimumProcessRuntime)	Minimum process runtime
ePR (MeanProcessRunTime)	Average process runtime
aPR (MaximumProcessRunTime)	Maximum process runtime
NA (NumberOfActivities)	Total number of all activities of running processes
NAF (NumberOfActivitiesFinished)	Number of finished activities
NAE (NumberOfActivitiesError)	Number of erroneous activities
NAS (NumberOfActivitiesPause)	Number of paused activities
NAU (NumberOfActivitiesUnfinished)	Number of not finished activities
NAP (NumberOfActivitiesPersonalized)	Number of personalized activities
sA (EscalationsActivities)	Total number of escalations (expired deadlines) of all activities
sF (EscalationsActivitiesFinished)	Total number of escalations (expired deadlines) of finished activities
sU (EscalationsActivitiesUnfinished)	Total number of escalations (expired deadlines) of unfinished activities
sAP (EscalationsActivitiesPersonalized)	Total number of escalations (expired deadlines) of personalized activities
iAR (MinActivityRunTime)	Minimum activity runtime
iAF (MinActivityFinishedRunTime)	Minimum runtime of finished activities
iAU (MinActivityUnfinishedRunTime)	Minimum runtime of unfinished activities
iAP (MinActivityPersonalizedRunTime)	Minimum runtime of personalized activities
eAR (MeanActivityRunTime)	Average activity runtime
eAF (MeanActivityFinishedRunTime)	Average runtime of finished activities

eAU (MeanActivityUnfinishedRunTime)	Average runtime of unfinished activities
eAP (MeanActivityPersonalizedRunTime)	Average runtime of personalized activities
aAR (MaxActivityRunTime)	Maximum activity runtime
aAF (MaxActivityFinishedRunTime)	Maximum runtime of finished activities
aAU (MaxActivityUnfinishedRunTime)	Maximum runtime of unfinished activities
aAP (MaxActivityPersonalizedRunTime)	Maximum runtime of personalized activities

AD Tag (ActivityData), AE Tag (ActivityDataEntry)

The AD tag is included in the D tag, groups all AE tags and has no attributes. The AD tag only exists after being specified in the configuration by the 'DrillDown' type. The AE tag provides information on an activity and has the following attributes (if the corresponding options are activated in the report configuration):

FI (FamilyId)	Family identification
FN (FamilyName)	Family name
AI (ActivityId)	Activity identification in the model
AN (ActivityName)	Activity name
NA (NumberOfActivities)	Total number of all activities of running processes
NAF (NumberOfActivitiesFinished)	Number of finished activities
NAE (NumberOfActivitiesError)	Number of erroneous activities
NAS (NumberOfActivitiesPause)	Number of paused activities
NAU (NumberOfActivitiesUnfinished)	Number of not finished activities
NAP (NumberOfActivitiesPersonalized)	Number of personalized activities
sA (EscalationsActivities)	Total number of escalations (expired deadlines) of all activities
sF (EscalationsActivitiesFinished)	Total number of escalations (expired deadlines) of finished activities
sU (EscalationsActivitiesUnfinished)	Total number of escalations (expired deadlines) of unfinished activities
sAP (EscalationsActivitiesPersonalized)	Total number of escalations (expired deadlines) of personalized activities
iAR (MinActivityRunTime)	Minimum activity runtime
iAF (MinActivityFinishedRunTime)	Minimum runtime of finished activities
iAU (MinActivityUnfinishedRunTime)	Minimum runtime of unfinished activities
iAP (MinActivityPersonalizedRunTime)	Minimum runtime of personalized activities
eAR (MeanActivityRunTime)	Average activity runtime
eAF (MeanActivityFinishedRunTime)	Average runtime of finished activities
eAU (MeanActivityUnfinishedRunTime)	Average runtime of unfinished activities
eAP (MeanActivityPersonalizedRunTime)	Average runtime of personalized activities
aAR (MaxActivityRunTime)	Maximum activity runtime
aAF (MaxActivityFinishedRunTime)	Maximum runtime of finished activities
aAU (MaxActivityUnfinishedRunTime)	Maximum runtime of unfinished activities
aAP (MaxActivityPersonalizedRunTime)	Maximum runtime of personalized activities

Detailed Process Report

D Tag (Data) in SR Tag (SR/D)

The D tag is included in the SR tag and has no attributes. It always contains a Ds tag and a P tag. Depending on the report configuration it can also contain an AD tag.

Ds Tag (DataFields) in D Tag (SR/D/DS)

The Ds tag groups D tags and has no attributes. A D tag in a Ds tag assigns identifications of process variables, which may be in the report, to names which are supposed be used for display. The following attributes are available for a D tag in a Ds tag:

FI (FamilyId)	Family identification in which the name is supposed to be used
Id	Process variable identification
N (Name)	Name of the process variable to be displayed

P Tag (ProcessData), E Tag (ProcessEntry) (SR/D/P/E)

The P tag is included in the SR tag and has no attributes. It contains an E tag for each running process. E tags can contain Ds tags and have the following attributes (if the corresponding options are activated in the report configuration):

Id	Process identification
N (Name)	Process name
S (Subject)	Process subject
FI (FamilyId)	Family identification
FN (FamilyName)	Family name
MI (ModelId)	Model identification
MN (ModelName)	Model name
C (CreationTime)	Time stamp of process creation
CY (CreationYear)	Year of process creation
CM (CreationMonth)	Month of process creation
CD (CreationDay)	Day of process creation
CH (CreationHour)	Hour of process creation
CU (CreationMinute)	Minute of process creation
CS (CreationSecond)	Second of process creation
R (RunTime)	Running time of the process
St (State)	Process status: 0x00 unknown 0x01 initialized 0x02 running 0x04 stopped 0x08 active 0x10 canceled 0x20 finished 0x40 erroneous, stopped by system
NA (NumberOfActivities)	Total number of all process activities
NAF (NumberOfActivitiesFinished)	Number of finished activities
NAU (NumberOfActivitiesUnfinished)	Number of not finished activities
NAP (NumberOfActivitiesPersonalized)	Number of personalized activities

NAS (NumberOfActivitiesPaused)	Number of paused activities
NAE (NumberOfActivitiesError)	Number of erroneous activities
sP (EscalationsProcess)	Total number of process escalations (expired deadlines)
sA (EscalationsActivity)	Total number of escalations (expired deadlines) of all process activities
sU (EscalationsActivitiesUnfinished)	Total number of escalations (expired deadlines) of unfinished activities
sF (EscalationsActivitiesFinished)	Total number of escalations (expired deadlines) of finished activities
sAP (EscalationsActivitiesPersonalized)	Total number of escalations (expired deadlines) of personalized activities

Ds Tag (DataFields) in P Tag (SR/D/P/Ds/D)

The Ds tag groups D tags and has no attributes. A D tag contains values of process variables and has the following attribute:

Id	Process variable identification
----	---------------------------------

Depending on the type of the process variable, the D tag can contain another tag that in turn has the value of the process variable. The following tags are allowed within a D tag:

S (String)	For strings
B (Boolean)	For logical values
L (Long)	For integer values
F (Float)	For floating point numbers
D (Date)	For date specifications

AD Tag (ActivityDate), AE Tag (ActivityDataEntry) (SR/D/AD/AE)

The AD tag is included in the D tag, groups all AE tags and has no attributes. The AD tag only exists after being specified in the configuration by the 'DrillDown' type. The AE tag provides information on a process activity and has the following attributes (if the corresponding options are activated in the report configuration):

Id (ProcessId)	Process identification
AI (ActivityId)	Activity identification
MA (ModelActivityId)	Activity identification in the model
AN (ActivityName)	Activity name
T (StartTime)	Start time (time stamp)
TY (StartYear)	Year of the start time
TM (StartMonth)	Month of the start time
TD (StartDay)	Day of the start time
TH (StartHour)	Hour of the start time
TU (StartMinute)	Minute of the start time
TS (StartSecond)	Second of the start time
R (RunTime)	Runtime

S (State)	Status: 0x00000000 unknown 0x00000001 created 0x00000002 started 0x00000004 StartActivityEvent executed 0x00000008 EndActivityEvent executed 0x00000010 loop condition verified 0x00000020 execute loop body 0x00000040 Process step created 0x00000080 Process step personalized 0x00000100 Waiting for retention period 0x00000200 Waiting for subflow 0x00000400 executed 0x00000800 forwarded 0x00001000 finished 0x00002000 stopped 0x00004000 finished 0x10000000 erroneous
E (NumberOfEscalations)	Number of escalations (expired deadlines)