Together Workflow Editor

V.4.4-1 - 20120111-0100-TAB-1.4-2

User Manual

Together Teamsolutions Co., Ltd. in Thailand.

Together Workflow Editor: User Manual

by Together Teamsolutions Co., Ltd. in Thailand.

In this document parts of the original XPDL 2.1 specification - WfMC XPDL 2.1 Specification[1] are used. The Workflow Management Coalition and its members are the owners of the copyright of this specification.

---

[1] http://www.wfmc.org/Download-document/WFMC-TC-1025-Oct-10-08-A-Final-XPDL-2.1-Specification.html

Table of Contents

List of Figures

List of Tables

Preface

Together Workflow Editor[1] is a visual tool for creating, managing and reviewing process definitions stored in WfMC XPDL syntax, using BPMN graphical notation.

Put simple, Together Workflow Editor allows you to quickly create or view XPDL workflow definition files, check and store them for the further use. Once a definition is proven valid, it can be referenced by new definitions, thus shortening the time and effort needed to define workflow processes or imported into workflow engines capable of XPDL for execution.

About Workflow

From the Workflow Management Coalition website[2]:

Why Should a Business Use Workflow ?

The Evolution of Workflow Workflow Management consists of the automation of business procedures or "workflows" during which documents, information or tasks are passed from one participant to another in a way that is governed by rules or procedures. Workflow software products, like other software technologies, have evolved from diverse origins. While some offerings have been developed as pure workflow software, many have evolved from image management systems, document management systems, relational or object database systems, and electronic mail systems.

The Key Benefits

- Improved efficiency - automation of many business processes results in the elimination of many unnecessary steps
- Better process control - improved management of business processes achieved through standardizing working methods and the availability of audit trails
- Improved customer service - consistency in the processes leads to greater predictability in levels of response to customers
- Flexibility - software control over processes enables their re-design in line with changing business needs
- Business process improvement - focus on business processes leads to their streamlining and simplification

Open source

Together Workflow Editor is available as an Open Source as the Enhydra JaWE (Java Workflow Editor) project under the GPL V3 license.

Enhydra JaWE[3] is an open source[4] project hosted by SourceForge[5] , with the goal to deliver a production quality tool for graphical creation and viewing of WfMC XPDL process definitions. It results from real world projects in which the painful task of writing XML with general purpose XML editors had to be done again and again. This project is developed mainly by employees of Together Teamsolutions Co., Ltd[6] but also contains lots of input and improvements

---

[1] http://www.together.at/prod/workflow/twe
[2] http://www.wfmc.org
[3] http://sourceforge.net/projects/jawe
[4] http://www.opensource.org
[5] http://sourceforge.net
[6] http://www.together.at

from the many users worldwide. Enhydra JaWE has a rapidly growing community and is also integrated in other open source projects like Enhydra Server[7] - a professional Java and J2EE application server and Enhydra Shark[8] - the most flexible Open Source WfMC XPDL workflow engine available. Its use is "Free as in Freedom"[9] and licensed under the GNU [10]GPL V3 license[11] defined by the Free Software Foundation FSF[12].

Commercial Support

Commercial support for Together Workflow Editor including development of project specific extensions or integration into other environments and projects is offered by Together Teamsolutions Co., Ltd located in Pattaya/Thailand/Asia (mailto: office@together.at[13]).

Why read this book ?

This book is designed to be the clear, concise explanation and reference to Together Workflow Editor and its usage to graphically design and view WfMC XPDL process definitions.

We hope to answer all the questions that you might have about all the various XPDL elements, their meanings and how to view and edit them with Together Workflow Editor. In particular, this covers the following subjects:

- The structure of the editor and its components.
- Editing dialogs and property panels for all XPDL elements.
- The relationship between XPDL and Together Workflow Editor including in depth explanantions of all XPDL elements.
- Information how to configure and customize Together Workflow Editor to your needs.
- A step by step introduction how to build XPDL definitions from scratch.

This book's audience

We expect that most readers will have some familiarity with XML and workflow. Even if you just know what processes and activities are, you’re in good shape. Although this book provides basic introductions to key concepts, it may not suffice as your only tutorial about XPDL, XML, and workflow systems.

Readers of this book are typically developers, workflow modelers and workflow system administrators.

Some sections in this book describe certain environments like Microsoft Windows or Linux, although there is nothing about Together Workflow Editor that makes it unsuitable for the Mac or any other environment of your choice, as long as the defined software prerequisites (basically a JRE - Java Runtime Environment 1.4) are met.

Organization of this book

This book is divided into the following chapters:

---

[7] http://www.together.at/prod/server/tas
[8] http://sourceforge.net/projects/sharkwf
[9] http://www.gnu.org/philosophy/philosophy.html
[10] http://www.gnu.org
[11] http://www.gnu.org/copyleft/gpl.html
[12] http://www.gnu.org/fsf/fsf.html
[13] mailto: office@together.at

1. Introduction - What is Together Workflow Editor all about ?
2. Build Guide - How to compile, build and package your own version of TWE from the source code
3. Installation Guide - How to install TWE
4. User Interface - Explanation of the user interface components, menus and toolbars
5. Dialogs and Property Panels - How to handle editor dialogs and XPDL property panels
6. XPDL Elements - What they are, how they are related and how to create, edit and view them
7. XPDL from Scratch - How to create your first XPDL file
8. Configuration - How to tailor the Editor to your needs
9. Customization - How to extend the Editor with additional functionality
10. Extended Attribute Reference - XPDL Extended Attributes used by the Editor itself
11. Release notes - Notes about the major changes in each release

Request for Comments

Please help us to improve future editions of this book by reporting any errors, inaccuracies, bugs, misleading or confusing statements, and plain old typos that you find. Email your findings to twe@togetherteam.co.th[14]. Thanks.

Acknowledgements

Many thanks go to the Swiss company ABACUS[15] for sponsoring major parts of the Together Workflow Editor Version 2.0 development.

About the Authors

All authors are directly involved in one way or the other into the core development of Together Workflow Editor and Enhydra JaWE.

Together Teamsolutions Co., Ltd

Together Teamsolutions Co., Ltd has been active for many years primarily in the area of workflow, database and document management solutions.

In addition to the organizational and implementation services, these application areas are supplemented by the operation of solutions (system rollout, 24x7 hotline, training) as well as product development, consulting services (business process analysis and modelling), publications and presentations. The associated topics needed for real world projects, e.g. imaging, OCR, host integration, archiving, computer telephone integration, digital signatures, backup, performance tuning, reporting, systems administration and hardware sizing are also covered, continuously maintained and expanded.

Together Teamsolutions Co., Ltd is directly involved into the development of the XPDL standards specification.

---

[14] mailto:twe@together.at
[15] http://www.abacus.ch

Enhydra

The Enhydra.org project has its focus on E-Business middleware software. It now consists of many developers from over 50 countries. Many of these are able to offer commercial products, support, consulting and training.

Enhydra.org started with Enhydra, the first and leading Open Source Java/XML application server. It was initially created by Lutris Technologies, Inc. which already disappeared from the market. After 4 years in development it was open sourced on Janurary 15th 1999. They named the technology "Enhydra" after the California sea otter (Enhydra Lutris) - a popular inhabitant of Santa Cruz.

Big parts of Enhydra development are now sponsored by Together Teamsolutions Co., Ltd located in Thailand. One of the outcomes of this sponsorship is Enhydra JaWE / Together Workflow Editor.

JaWE

JaWE - Java Workflow Editor - is the Open Source project of Together Workflow Editor. It is fully useable to edit, view and save complete XPDL files and licensed under GPL V3. Sources are publicly available in the projects CVS repository hosted at SourceForge

WfMC - The Workflow Management Coalition

Founded in August 1993, the Workflow Management Coalition (WfMC) is a non-profit, global organization of adopters, developers, consultants, analysts and university/research groups engaged in Business Process Management (BPM). Workflow or BPM is the technology that supports people to do their work. Resulting benefit is more efficiency, traceability,and quality improvements.

WfMC creates and contributes to process related standards and educates on the benefits of process automation. WfMC is the only standards organization that concentrates purely on process. WfMC created XPDL and Wf-XML and had influence on BPMN, OMG workflow interface, ASAP and many other process related standards.

JGraph

JGraph[16] is a powerful, easy-to-use, feature-rich and standards-compliant open source graph component available for Java. This library is used in Together Workflow Editor for graphical representation of process layouts (graph component).

Docbook

This book was written using DocBook[17] and the XML Mind XML Editor[18].

---

[16] http://www.jgraph.com
[17] http://www.docbook.org
[18] http://www.xmlmind.com/xmleditor

Chapter 1. Introduction

What is Together Workflow Editor ?

Together Workflow Editor (TWE) is a graphical editor for creating, editing, managing and reviewing WfMC XPDL process definition files.

Workflow management is an evolving technology with lots of vendors claiming their approach is the best. We have chosen an approach which relies on WfMC and XPDL, thus supporting efforts to establish the standard. Together Workflow Editor is based on version 2.1 of the XPDL XML schema published by WfMC.

The editor makes creating and editing XPDL easy. It represents all XPDL elements graphically through property panels and a graph component using BPMN graphical notation, to give the user a better understanding and an overview of the process definitions. Various functions help in finding specific activities, participants, applications, errors in the model, etc.

The final output of the editor is an XML file (using the standardized WfMC XPDL schema) which can then be interpreted and executed by all WfMC XPDL compliant workflow engines.

Together Workflow Editor accomplishes three main goals:

- Reading of WfMC XPDL files (no matter from which tool they initially come from) from the filesystem or from the Wf-XML server
- Graphical representation and guided editing/modelling of process definitions
- Writing WfMC XPDL process definition XML files to the filesystem or to the Wf-XML server

History, Presence and Future Goals

Before version 4.x, Together Workflow Editor was XPDL 1 editor that coverd the whole XPDL 1 Meta-Model. It was the best open-source (and we believe also the best of all) editor which allowed any vendor to create the XPDL 1 based workflow processes for their workflow engines. It was used by many vendors to (graphically) design their process definitions. Beside that, it was always the example for other implementors of such editors to see how to correctly interpret XPDL specification.

Beside the core features that allows anyone to easily model XPDL processes, the flexibility and configurability of the editor, that includes configuration by only changing the existing component's properties, branding, and writting a special "configuration" modes for an editor, makes TWE powerful, and easily adjustable for the usage with any specific workflow engine.

The version 4.x brings XPDL 2 and BPMN support. Although not fully covering XPDL 2 and BPMN, TWE now allows users to use the old powers of the editor and new BPMN notation which goal is to standardize graphical process representations, and thus making them uniform accross different editor tools, and understandable by the users knowing BPMN standard.

The great feature of this new version is that it automatically converts XPDL 1 process definitions into XPDL 2 definitions, which minimizes the efforts of the migration to XPDL 2. Editor currently supports the sub-set of XPDL 2 specification necessary to conform to "simple" BPMN requirements.

In the future, editor will support more and more of BPMN and XPDL 2 specification, but we will always make our focus to first implement things that provide the modeler writting the workflow definitions for the real life human workflow scenarios.

Useful links

The WfMC Workflow Management Coalition[1] promotes the advancement of process management technology standards and their use by the industry.

The Java 2 SDK or Java 2 JRE can be downloaded from http://www.oracle.com/technetwork/java .

The Together Workflow Editor home page is located at http://www.together.at/prod/workflow/twe.

JGraph[2] is an open-source swing graph component used in Together Workflow Editor.

Enhydra Shark[3] - the most flexible Open Source WfMC XPDL workflow engine available

---

[1] http://www.wfmc.org
[2] http://www.jgraph.com
[3] http://sourceforge.net/projects/sharkwf

Chapter 2. Build Guide

By the end of this chapter you will know the prerequisites for installation, how and where to download the latest release binaries and source codes of Together Workflow Editor, how to install it on MS Windows or Linux and be able to use the provided XPDL examples.

Getting the source code

The source code of the project can be obtained either via SVN (please read instructions how to check out sources at SourceForge[1]) or by downloading the latest twe-x.y-z.src.zip/twe-x.y-z.src.tar.gz package from SourceForge[2].

Prerequisites

- Windows
  - Java Development Kit (JDK) version 6 or later
- Fedora (Linux)
  - bash
  - tar
  - make
  - rpm-build
  - Java Development Kit (JDK) version 6 or later

Preparing the build environment

Execute the configure script from the root directory of the project source. Specific JAVA version can be set for building (different from the one registered with your system) by executing:

- Windows

  configure -jdkhome %JAVA_HOME%

- Fedora (Linux)

  ./configure -jdkhome %JAVA_HOME%

(Where JAVA_HOME is the path to your JDK installation)

Possible parameters for the configure script are:

```
configure             - Creates build.properties file with default values
                        for all possible parameters. It can work only if
                        there is a default JAVA registered with the system.
configure -help       - Displays Help screen
```

---

[1] http://sourceforge.net/projects/jawe/develop
[2] http://sourceforge.net/projects/jawe/files

```
configure -version    - Sets the version number for the project.
configure -release    - Sets the release number for the project.
configure -buildid    - Sets the build id for the project.
configure -jdkhome    - Sets the "JAVA HOME" location of Java to be used
                        to compile the project.
configure -instdir    - Sets the location of the installation dir used when
                        executing make script with install target specified.
configure -rebranding - ONLY FOR WINDOWS. Flag that determines if the project
                        will be "rebranded" with the context of branding folder.
                        Possible values [true/false].
configure -language   - ONLY FOR WINDOWS. Used by NSIS when creating setup
                        (normally used for rebranding).
                        Possible values [English/Portuguese/PortugueseBR].

Multiple parameters can be specified at once.


Example:

configure -version 3.3 -release 1 -buildid 20110721-0808 -jdkhome C:/jdk1.6 -instdir C:/Ja
```

The configure script will create/change the build.properties file based on the parameters provided. This file can also be manually changed to adjust your environment/parameters for building the project from the sources.

Compiling and building

Execute the make script with the buildAll target from the root directory of the project source. When the building process finishes, the project binaries will be in output/twe-[version]-[release] folder.

Possible build targets for the make script are:

```
make help            - Displays Help screen
make buildAll        - Builds and configures TWE with documentation
make buildNoDoc      - Builds and configures TWE without documentation
make buildDoc        - Builds documentation only
make debug           - Builds TWE JAR files with included debug information
make install         - Installs and configures TWE into directory defined
                       by parameter install.dir in build.properties file.
                       This parameter can be set by using command:
                       configure -instdir PATH_TO_DIR.
                       It should be called only after make buildAll target
                       is executed!
make clean           - Removes the output and distribution folder (in order
                       to start a new compilation from scratch)
make distributions   - Builds and configures TWE with all documentations
                       and creates distribution package
```

Packaging distributions

Assuming that the environment is already configured as described previously, to create the project distribution packages, execute:

```
make distributions
```

When the building process finishes, the **distribution** folder will be created in the root directory of the project source containing the appropriate OS specific binary distributions.

On Windows, to have the resulting ‘.exe’ file automatically signed, the file called ‘sign.properties’ should be placed in the root directory of the projects source with the following properties:

```
sign.tool       - absolute path to the sign tool executable file
sign.privatekey - absolute path to the private key used for signing
sign.pwd        - password for signing
sign.alias      - sign alias
```

example sign.properties file:

```
sign.tool=D:/signtool/signtool.exe
sign.privatekey=D:/signtool/privatekey.pfx
sign.pwd=agles87t24e25NDwas
sign.alias=pvktmp:3852567a-45er-567y-w456-23456789sdft
```

Sign alias can be produced by the usage of Java's keytool by executing the command:

keytool -v -list -storetype pkcs12 -storepass PRIVATE-KEY-PASSWORD -keystore PATH-TO-PRIVATE-KEY

where PRIVATE-KEY-PASSWORD is the same as the property sign.pwd from sign.properties file described above, and PATH-TO-PRIVATE-KEY is the same as the property sign.privatekey from this properties file.

Rebranding

TWE build procedure enables you to create so called "rebranded" version of TWE distribution under Windows.

It means that at the end you can get the windows setup file fully "rebranded" as if the product is under your own ownership. E.g. instead of calling the product "Together Workflow Editor", you can call it "XYZ Workflow Editor", and during the build procedure replace all other things neccessary for the "rebranding".

To build rebranded product, first you have to configure TWE by executing the following configure command in %TWE_HOME% folder:

```
configure -rebranding true
```

If you also want to use a different language in the windows setup wizard, you should execute e.g. the following:

```
configure -language Portuguese
```

NOTE: currently possible values are English, Portuguese and PortugueseBR.

After that, several things needs to be done in the %TWE_HOME%/branding folder:

There are several sub-folders in %TWE_HOME%/branding folder which content needs to be edited,removed or appended in order to rebrand the application distribution. The following table explains the meaning of the sub-folders and how can you perform TWE branding by changing their content:

Table 2.1. Explanation for %TWE_HOME%/branding

Branding sub-directory	Description
aboutbox	Edit aboutbox.properties file to define what will be shown in the TWE's aboutbox If you don't want to show license information in the aboutbox, change the value of showLicenseInfo property to false.

Branding sub-directory	Description
	Example aboutbox.properties file is put here to show you how to do it - it specifies XYZ editor instead of TWE editor.
activityicons	If you put any icons into this folder, these will be the icons you will be able to chose for the "Icon" property from Activity's property panel. If you don't put any icons here, the default set of icons will be used from this panel. 6 sample icons are added here to show the use case.
config	If you want to change some of the existing TWE configuration modes, you should add new folder into this sub-folder that would contain modified TWE property files. Also, 'defaultconfig' file in the root of config folder specify default configuration to use during TWE startup. Example file in-here shows how to set default startup configuration to 'default' and contains 2 configuration files which are modified (comparing to original ones): a) towebasic.properties: - the default language is set to Portuguese - configured not to have default transient packages b) togwegraphcontroller.properties - configured to show grid - configured to show transition conditions - configured to show text at right of the activity box - changed graph background color - changed graph grid color c) jawetypes.properties - configured to use 2 custom activities
doc	If you want to change documentation and pictures appearing in the documentation, in this folder you should put the modified twe-doc.xml file and in Images sub-folder you should put all the pictures from the original folder you want to override. Example twe-doc.xml file is a very short version of standard User Manual, and changes all the occurences of TWE with XYZ, all occurences of Together Workflow Editor with XYZ Workflow Editor, all occurences of Together Teamsolutions Co., Ltd. with XYZ Company, all the URLs to together site with URLs to google site, etc.

Branding sub-directory	Description
	Example images folder provides a single images to override - the image in its original form contains TWE logo. For the list of all the re-placable images you should look into %TWE_HOME%\doc\Images folder
examples	If you put anything in this folder, the original examples folder will be replaced by this one. Example shows one XPDL file which normally is not the part of TWE distribution.
i18n	If you put language property file(s) here, it will override the original one, or will add one if it does not exist in the original distribution. Example provides JaWE_pt.properties file where all the occurences of TWE and Together are replaced by XYZ
images	If you put image here, it will override the original image coming with distribution. Example images put into this folder are the ones for the splash screen, jawe frame icon and icon for the generic activity. For the list of all the replacable images you should look at folder: %TWE_HOME%\modules\Utilities\src\org\enhydra\jawe\images
installation	Here you can put your own images and language files for the TWE installer. Example shows the names of the images you can replace and modification of Brazilian Portuguese language/branding file which refers to the editor as to XYZ Editor.
lib	If you need to add additional jar to be used, or you want to override an existing one, put it in this folder.
license	In this folder you put your own license. This license can appear in the about dialog if you set proper configuration switch in aboutbox.properties file (read remarks for aboutbox folder) and in the root folder of editor binary distribution. Example contains dummy license file with XYZ license.
registry	This folder contains icons that will be used to register application in windows registry. Example shows icon for XPDL file extension and icon for the editor itself.

After modifying the content of branding folders, simply continue with normal distribution packaging procedure, and the resulting output files in distribution folder will be rebranded.

Chapter 3. Installation guide

Getting the binaries

The latest binary packages of Together Workflow Editor can be downloaded from SourceForge[1]

Prerequisites

The only prerequisite to be able to run TWE on Windows or Linux system is Java JRE 1.6 installed on the machine.

Installation

There are several binary packages for Windows and Linux operating systems that can be used to install TWE.

If TWE is installed on Windows using **twe-x.y-z.exe** package, just follow the setup procedure. The similar is with **twe-x.y-z.noarch.rpm** package on Linux.

If TWE is installed from **twe-x.y-z.zip** or **twe-x.y-z.tar.gz** package on Windows/Linux respectivly, the packages should be un-packed, and in the root of the folder where they are unpacked, configure script should be executed. If JAVA_HOME environment variable exists, configure script can be executed without parameters, otherwise it should be called with a parameter specifying JAVA_HOME value, e.g:

configure -jdkhome c:\jdk1.6.0_20

This will create proper **run** script in bin directory that should be used to run TWE.

Silent Installation

TWE has possibility to silently install via **twe-x.y-z.exe** file. To do that, rename the **twe-x.y-z.silent.properties.txt** file comming with TWE distribution into **twe-x.y-z.silent.properties**, put it into the same folder with the **twe-x.y-z.exe**, and normally start the installation. During the installation, there will be no dialogs asking you to chose Java, place to install, etc. This information is taken from **twe-x.y-z.silent.properties** file. Here is the content of that file, with the properties you should modify to customize your installation:

```
# Where to install Together Workflow Editor (default value is C:\Program Files\twe-x.y-z)
inst.dir=C:\Program Files\twe-4.1-1

# Path to local java installation (obligated - has no default value)
jdk.dir=C:\Program Files\Java\jdk1.6.0_24

# Startup menu name. (default value - Together Workflow Editor x.y-z)
startup.menu.name=Together Workflow Editor 4.1-1

# Create quick launch icon (on/off)
create.quick.launch.icon=on

# Create start menu entry (on/off)
create.start.menu.entry=on
```

---

[1] http://sourceforge.net/projects/jawe/files

```
# Create desktop icon (on/off)
create.desktop.icon=on
```

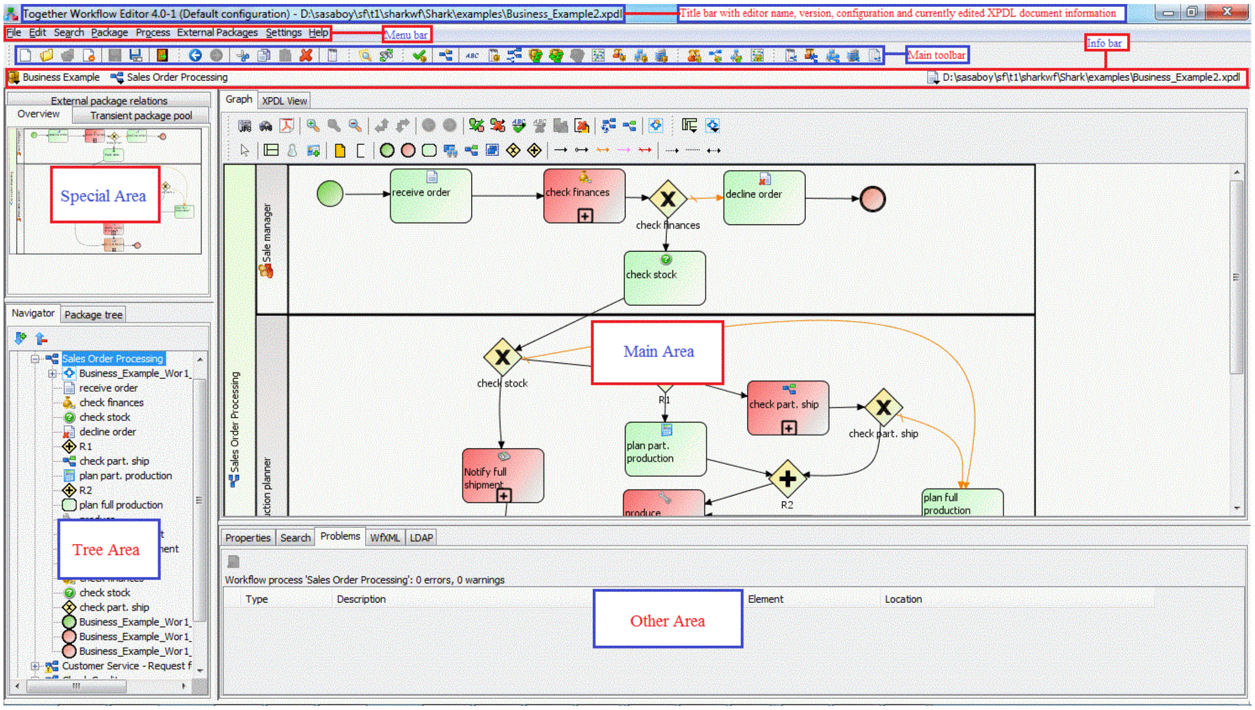
Chapter 4. User Interface

The Together Workflow Editor user interface consists of several parts. The order and the number of the displayed parts depend on the actually used configuration. This chapter explains the default configuration layout.

Right below the title line of the editor window you will see the main menu, the main tool bar and the info bar. The rest of the window contains different components. The components are placed in the corresponding are depending on their type.

Each component is responsible either for providing a different view of the WfMC XPDL elements, for displaying some additional information about XPDL elements (such as validation problems) or for displaying the result of some search.

Figure 4.1. Default configuration layout



Workflow process 'Sales Order Processing': 0 errors, 0 warnings

There are two areas on the left hand side: The upper area (marked as "Special Area") contains the *Graph overview*, the *External package relations* and the *Transient package pool* components. The lower area (marked as "Tree Area") contains the tree type components *Navigator* and *Package tree*.

There are also two areas on the right hand side: The upper area (marked as "Main Area") contains the *Graph* and the *XPDL View* components. The lower area (marked as "Other Area") contains the *Properties*, *Search*, *Problems*, *WfXML* and LDAP components.

(NOTE: by right-clicking on the tab you are able to close some component's view or to move it from one area to another)

See the section called “Property file *togwecontroller.properties*”[96] for details about the configuration of the main frame.

The rest of this chapter explains all these editor parts in detail.

Main menu

This is the core of the Together Workflow Editor. All the actions needed for opening, creating and manipulating XPDL files and elements are here.

The actions in the main menu are organized in the following groups:

1. File

New	This action creates a new "empty" XPDL package.
Open	This action opens an existing XPDL file. The file is chosen through the standard file selection dialog.
Reopen	This action reopens the currently edited XPDL file.
Close	This action closes the currently open XPDL package / file.
Save	This action saves the currently open XPDL package to the file. If no file name has been specified for the current package, the "Save As..." dialog will be shown.
Save As...	This action saves the currently open XPDL package into a file with a new filename and location specified through a standard file dialog.
Recent Files	This action opens a sub-menu with a listing of recently used files which can be re-opened.
Exit	This action exits the editor applciation. An appropriate dialog will appear if the current work hasn't been saved into a file.

2. Edit

Undo	This action reverts the last operation on the XPDL model. Every operation / change performed to the XPDL model can be undone step-by-step without any limits.
Redo	This action repeats the previously undone XPDL model operation.
Cut	This action will remove the currently selected elements from the model and place them into the clipboard, allowing the user to paste them somewhere else. This operation uses the internal clipboard of the editor. Any previous content of the internal clipboard is discarded when this action is executed.
Copy	This action will copy the currently selected elements into the clipboard, allowing the user to paste them somewhere else. This operation uses the internal clipboard of the editor. Any previous content of the internal clipboard is discarded when this action is executed.

Paste	This action will copy the elements from the internal clipboard to the current location. It will only be enabled if the internal clipboard currently contains at least one element.
Delete	This action removes the currently selected elements from the XPDL model.
Properties	This action opens a dialog to define properties for the currently selected element.

3. Search

Search	This action finds XPDL elements based on certain search criteria.
References	This action seaches for all elements in the XPDL model which refer to the currently selected element.

4. Package

Check validity	This action runs a check on the XPDL model's validity based on the WfMC specification and produces a problem list.
Insert new process	This action inserts a new workflow process definition into the XPDL package.
Namespaces	This action shows a dialog for handling the package's namespaces.
Package properties	This action opens a dialog for editing the XPDL package properties.
Processes	This action shows dialog with list of the workflow process definitions defined in selected XPDL package.
External packages	This action opens a dialog to handle references to external XPDL packages.
Add external package	This action opens a dialog to import / reference an external package.
Remove external package	This action removes an external package reference.
Type declarations	This action shows a dialog to manage type declarations.
Participants	This action shows a dialog to managing participants.
Applications	This action shows a dialog to manage applications.
Workflow variables	This action shows a dialog to manage variables.
Open referred document	This action opens the document attached to the package.

5. Process

Process properties	This action opens a dialog to edit the properties of the selected process.
Participants	This action shows a dialog to manage participants of the selected process.
Applications	This action shows a dialog to manage applications of the selected process.
Workflow variables	This action shows a dialog to manage variables of the selected process.
Formal parameters	This action shows a dialog to manage parameters of the selected process.
Activity sets	This action shows a list of all activity sets contained within the selected process.
Activities	This action shows a list of activities contained within the selected process.
Transitions	This action shows a list of transitions contained within the selected process.

6. External Packages

External participants	This action shows a list of all participants that are in the external packages.
External processes	This action shows a list of all processes that are in the external packages.
External applications	This action shows a list of all application that are in the external packages.
External type declarations	This action shows a list of all type declarations that are in the external packages.

7. Settings

Language	This menu shows the list of languages you can use (currently English, German, French, Portuguese and Serbian). The selection takes effect after you restart the editor.
Configuration	This menu shows the list of configurations that you can use (currently default, shark and samples-loopactivity configurations are provided). After you select the configuration, the editor is being re-configured to use this new configuration with all of its customizations (special validation, restrictions, etc.)

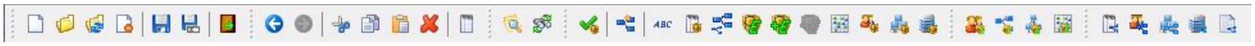
8. Help

Manual	This action opens the this User Guide.
About	This action shows the version, license and other information about the editor.

Main tool bar

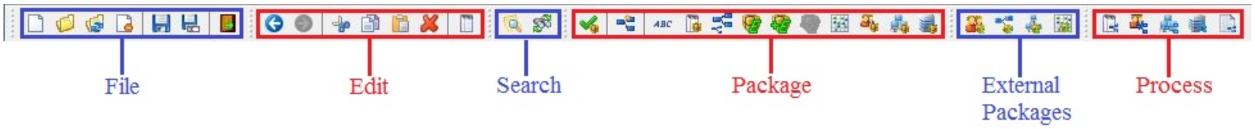
As explained in the previous section, the main menu covers all actions needed for creating, viewing and editing XPDL definitions. The Main tool bar contains shortcuts for various actions of the main menu.

Figure 4.2. Main tool bar



Main tool bar shortcuts are organized similar to the actions in the main menu. All actions are explained in the previous section.

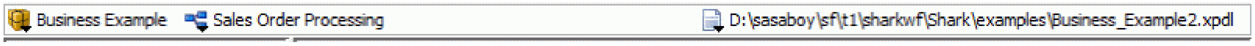
Figure 4.3. Main tool bar shortcut groups



Info bar

The Info bar shows some basic information about the selected workflow elements. It displays the package name, the process name and the file/package that is being edited.

Figure 4.4. Info bar



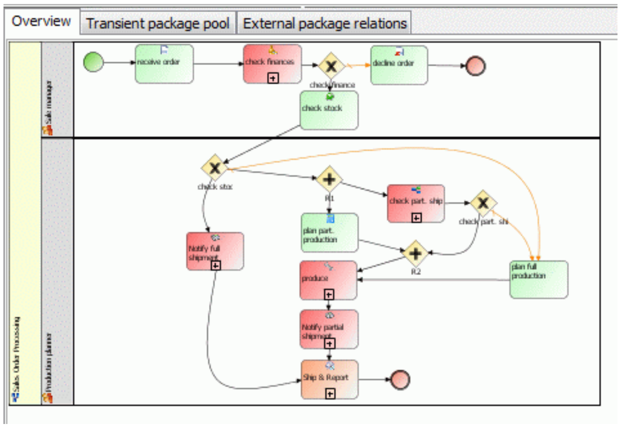
Business Example Sales Order Processing D:\sasaboy\sf\t1\sharkwf\Shark\examples\Business_Example2.xpdl

The Info bar offers the possibility to select the desired package, process or file:

- a click on the package icon offers a pop-up list of all packages (the main package you are editing and all the packages it references directly or indirectly).
- a click on the process icon offers a pop-up list of all processes (within the currently selected package).
- a click on the file icon offers a pop-up list of recently opened files.

Graph Overview

The Graph Overview is just what its name says - a graphical overview of the process and activity set graph. It shows the whole graph of the selected process or activity set in a way that you can see all the graph elements in the overview area at once.



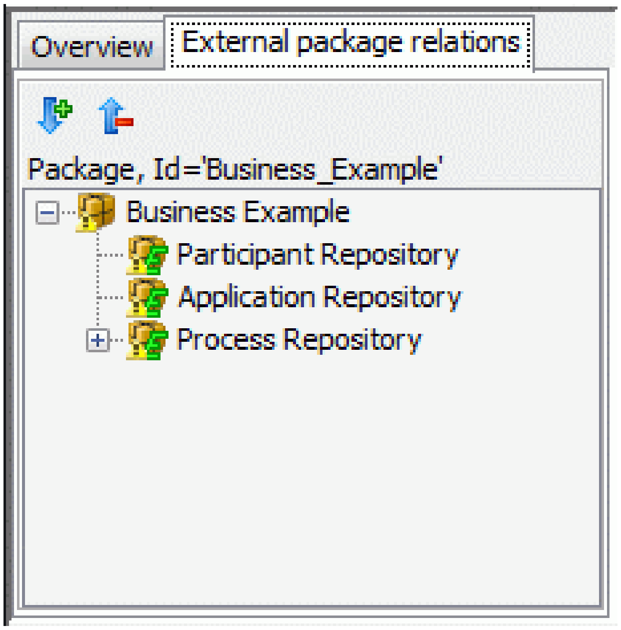
This overview also provides an easy way to select elements by clicking on the elements or dragging a selection rectangle in the small graphics. All selections done in the overview are immdiately reflected in the main graph. This way even bigger portions of the graph can be selected at once without having to scroll through the main graph.

If the element that is currently not visible in the main Graph is selected in the Graph Overview, the main graph is automatically scrolling to show the selected element thus making graphical navigation in large process graphs very easy.

The Graph Overview can be hidden by clicking on the "Overview" tab with the right mouse button and selecting "Close". It can be shown again by selecting "Add Overview" in the context menu of the "External package relations" tab or the background of the "Special Area".

External Package Relations

The External Package Relations component gives you a graphical tree overview of the current package and all of its directly or indirectly referrenced external packages. So it provides you the information about the relations between packages starting at the current package.



The picture on the left hand side shows a package (called "Business Example") that references three other external packages: "Participant Repository", "Application Respository" and "Process Repository". The external package "Process Repository" again references (the same) two external packages: "Participant Repository" and "Application Repository".

The "Expand All" and "Collapse All" buttons on top of the tree can be used to show or hide the subtree.

The External Package Relaltions can be hidden by clicking on the "Overview" tab with the right mouse button and selecting "Close". It can be shown again by selecting "Add External package relations" in the context menu of the "Overview" tab or the background of the "Special Area".

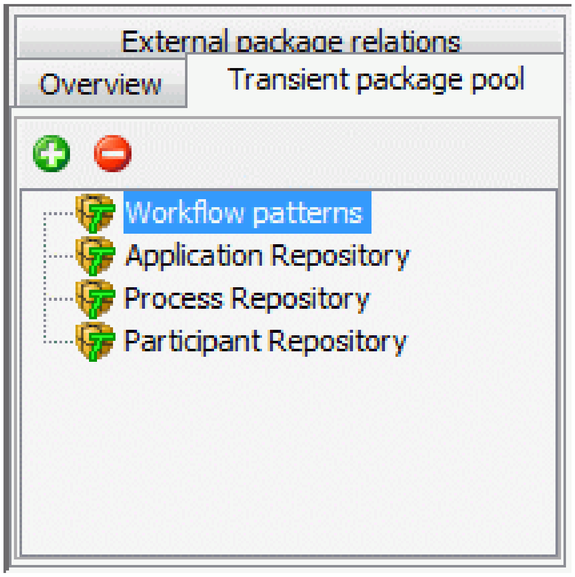
The External Package relations background color can be changed through the extpkgrelations.properties file. (BackgroundColor = R=?,G=?,B=?)

As with all other components you are also able to change the content of the toolbar and the order of the toolbar buttons.

Transient Package Pool

Transient packages are not related to any part of the WfMC XPDL specification but are a unique feature of Together Workflow Editor.

Transient packages are always available to the editor user (independently of the main package that is currently being edited) and the editor user can always copy elements (such as commonly used Activities and Transition structures, Applicatons or Participants definitions, etc.) from the transiently referrenced package into the currently edited package. XPDL Packages imported as transient are neither being validated for errors nor are these transient referrences stored in the current XPDL file. They are just shown for your convenience during modelling work. If a transient package is referencing other packages the user must be aware that the referenced packages won't be imported automatically.



The Transient Package Pool displays all the transiently opened packages.

Using the buttons on top you are able to add new or remove existing transiently referrenced packages from the editor.

The Transient Package Pool can be hidden by clicking on the "Transient package pool" tab with the right mouse button and selecting "Close". It can be shown again by selecting "Add Transient package pool" in the context menu of the "Overview" tab or the background of the "Special Area".

You are able to pre-configure Together Workflow Editor to load certain XPDL packages in transient mode (see the section called “Property file *togwebasic.properties*”) at startup to have access to most commonly needed XPDL elements for copying them into the XPDL package you are editing. This feature is typically used for copying certain workflow design patterns from a pattern pool into processes. Predefined common workflow patterns can be found in the Together Workflow Editor installation directory "...\examples\valid\WorkflowPatterns\".

You should be aware that there is a major difference in reusing XPDL elements from external packages and copying XPDL elements from other (transiently loaded) XPDL packages. Common subflows, application or participant definitions should normally be reused by modelling externally referenced packages instead of copying definitions redundantly into other packages.

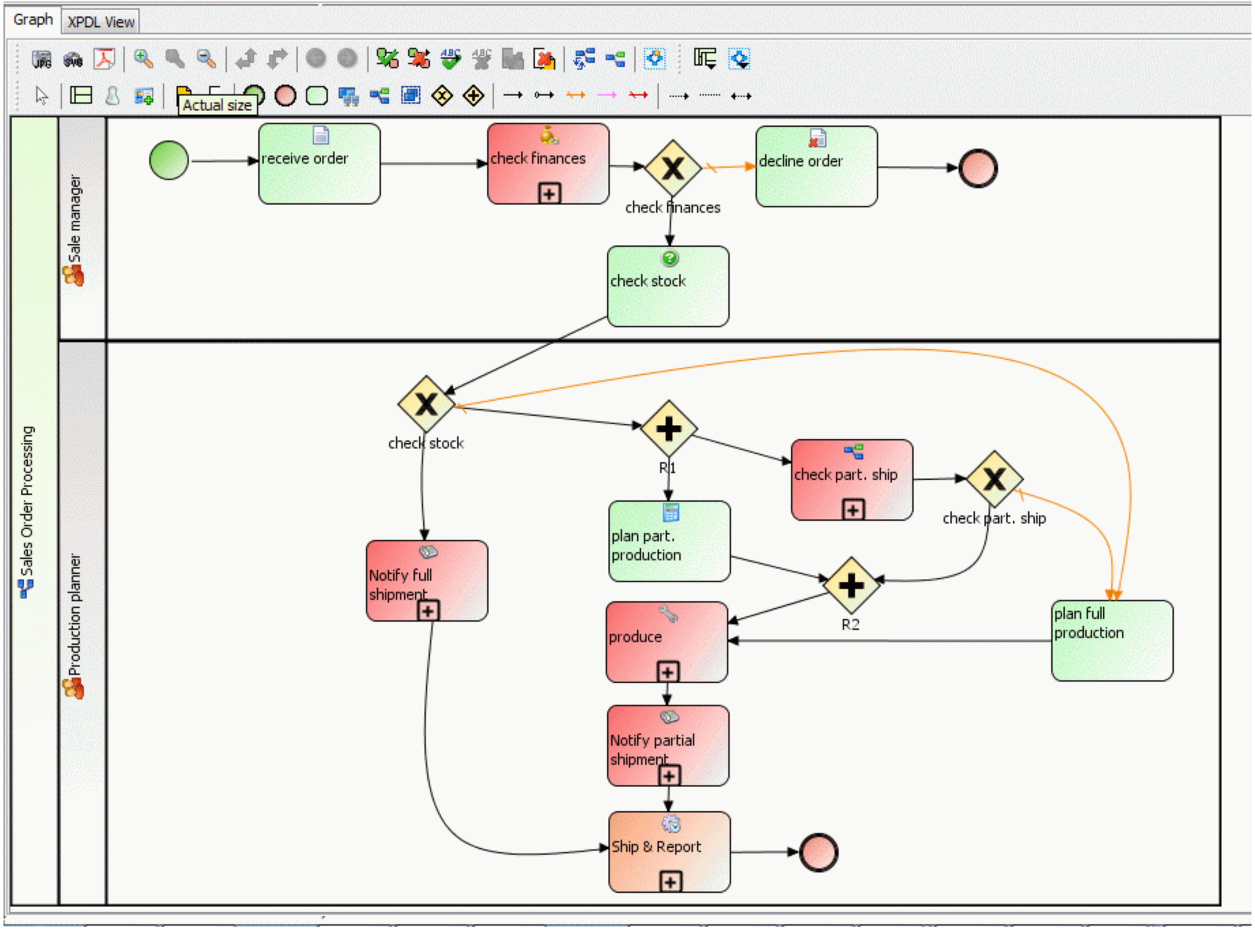
The Transient Package Pool background color can be changed through the transientpkgpool.properties file. (BackgroundColor = R=?,G=?,B=?)

As with all other components you are also able to change the content of the toolbar and the order of the toolbar buttons.

Graph

The Graph is the main component of Together Workflow Editor. It displays the graph representing the selected process or activity set. It offers the possibility to insert new elements into the graph and to visually define the flow / logic of the workflow process you are modelling.

Figure 4.5. Graph Panel



The Graph has its own toolbars with action Shortcuts to customize the view of the process, inserting new process elements or modify existing process elements.

The toolbar on top of the graph component consists of the following actions:

JPG	This action saves the graphical view of the current graph into a JPG format file.
SVG	This action saves the graphical view of the current graph into a SVG format file.
	This action saves the graphical view of the current graph and process information into a PDF format file.
🔍	This action zooms into the graph.
🔍	This action displays graph in its actual size.
🔍	This action zooms out of the graph.
⮤	This action moves the selected participant up or left depending on the current participant orientation of the graph.

	This action moves the selected participant down or right depending on the current participant orientation of the graph.
⬅	This action displays the part of the graph shown before the current part.
➡	This action displays the part of the graph shown after the current part.
✓	This action inserts the missing start and end events into the graph.
✗	This action removes all the start and end events from the graph.
ABC ✓	This action makes text describing transition condition to be shown in the graph.
ABC ✗	This action makes text describing transition condition to be hidden in the graph.
✓	This action shows artifacts (and associations) in the graph.
✗	This action hides artifacts (and associations) from the graph.
	This action changes the graph's swimlane orientation from left/right to top/bottom and vice versa.
	This action performs an automatic layout of the graph.
	This action inserts a new activity set.
	This action inserts a lane representing an existing participant into the graph.
	This action selects an existing activity set.

The toolbar below the one described above consists of the following actions:

↖	This action switches the mouse cursor to the selection mode.
	This action creates a new participant (on the package level) and inserts a new lane representing this participant into the graph.
	This action inserts a new "free text expression" lane into the graph.
	This action inserts a "common expression lane" into the graph.
🗎	This action inserts a data object artifact.
[	This action inserts a text annotation artifact.
●	This action inserts a start event activity.

●	This action inserts an end event activity.
▢	This action inserts an activity without implementation (manual activity performed by a human) into the graph.
▣	This action inserts a task-application activity into the graph
⧉	This action inserts a subflow activity into the graph.
⬚	This action inserts a block activity into the graph.
⊗	This action inserts a exclusive gateway - route activity into the graph.
⊕	This action inserts a parallel gateway - route activity into the graph.
◦→	This action inserts a conditional transition into the graph.
→	This action inserts an unconditional transition into the graph.
⇸	This action inserts an otherwise transition into the graph.
→	This action inserts an exception transition into the graph.
⇸	This action inserts a default exception transition into the graph.
⋯→	This action inserts a directional association into the graph.
⋯⋯	This action inserts non-directional association into the graph.
←⋯→	This action inserts bi-directional association into the graph.

The Graph provides all functions to handle lanes (representing participants), activities and transitions. All standard editor functions like insert, delete, move and select are supported.

When new Workflow Process is created, TWE automatically creates a Pool for that process, and generates a new Graph representing this workflow process. This graph contains only a swimlane representing Pool. The first thing that need to be added/drawn are the lanes (a swimlanes representing the participant). When at least one (swim)lane is visible you can add activities, artifacts, transitions and associations into the graph.

New elements are inserted into the graph in two steps. First select the button of choice in the graph toolbar to change the mouse cursor into the appropriate insert mode. Then click into the graph to insert the selected element. The mouse cursor will remain in the current insert mode until you click the right mouse button, press the ESC key on your keyboard or switch to another mode by selecting a different button in the graph toolbar.

Inserting transitions and associations into the graph is somewhat different: Transitions' begin and end points must be activities, and associations' begin/end points must be activity and artifact. Clicking on the empty background of the graph while inserting a transition/assocation will insert a graphical break/routepoint. If insertion of a transition/association is started and the right mouse button or the ESC key is pressed, the current insert operation is cancelled and the mouse cursor stays in transition/association insert mode.

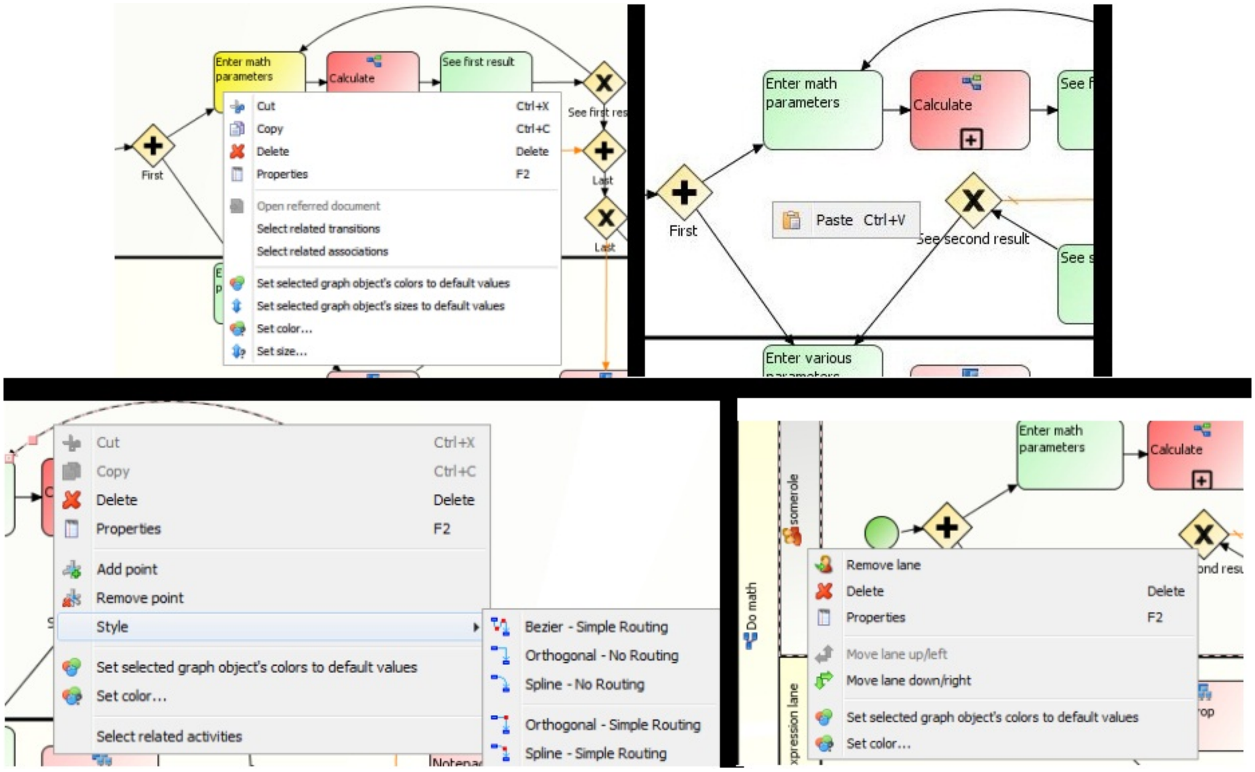
Elements in the graph are selected by simply clicking on them with the left mouse button. You can select a group of elements by clicking the left mouse button on the empty background of the graph and dragging a rectangle around some elements or by using the SHIFT or CTRL key during left mouse clicks on the individual elements.

To move elements to a new location just drag them with the left mouse button whereever you like.

Double clicking on a graph element opens the property dialog for the element (except for block and subflow activities for which the appropriate graph will be opened).

A right mouse click on an element in the graph will open the context popup menu of the element. The content of the context popup menu depends on which element was clicked on:

Figure 4.6. Graph element context popup menus



- The empty background represents the process itself and only the Paste action can be performed to copy elements from the internal clipboard to the selected location.

- Activities and Artifacts will show the standard edit functions. You can cut, copy, delete or edit properties of the selected activity/artifact, select related transitions/associations, or set color and size of the graphical object.

- Transitions and Associations will also allow cut/copy/paste actions but conditionally (if connecting activities/artifacts are selected). You can also add a new break point, remove an existing one, delete it, edit transition/association properties, change the transition/association style, execute action to select related activities/artifacts, or set color of graphical object

- Lanes can be removed, deleted, edited or repositioned in the graph. The color of the lane can be changed as well.

  Common expression lane (shown as special swimlane in the graph) can be associated with a new performer expression.

Be aware that there is a fundamental difference between removing a lane and deleting it:

- When lane is removed, it is removed from the selected process/activity set graph but it is still present in the XPDL model/package. However all the activities contained in the lane will be removed from the current graph.
- When lane is deleted, it is being deleted from the entire XPDL model/package but the activities contained in a lane won't be deleted from the model. They will be placed into the Graph's special lane called *Free text expression* lane.

As mentioned, when the graph is empty (e.g. when a new process is created, and only Pool for this process is displayed) the first thing to be added/drawn is a lane representing the participant. When at least one lane is visible activities, artifacts, transitions and associations can be inserted into the graph.

The available types of lanes are:

- Lane representing participant
- Free text expression lane
- Common expression lane

The available types of activities are:

- Start event activity (explicitly specifies process start)
- End event activity (explicitly specifies process end)
- Normal activity (manual activity without implementation)
- Task-application activity
- Sub-flow activity
- Block activity
- Exclusive gateway route activity
- Parallel gateway route activity

The available types of transitions are:

- Conditional transition
- Unconditional (uncontrolled flow) transition
- Otherwise (default flow) transition

- Exception transition
- Default exception transition

There are three association types available:

- directional association
- non-directional association
- bi-directional association

When inserting new lane into the graph using the toolbar buttons, the new participant on the package level is created, and its type is "Role". To change the participant type double click on the swimlane title to get its property panel.

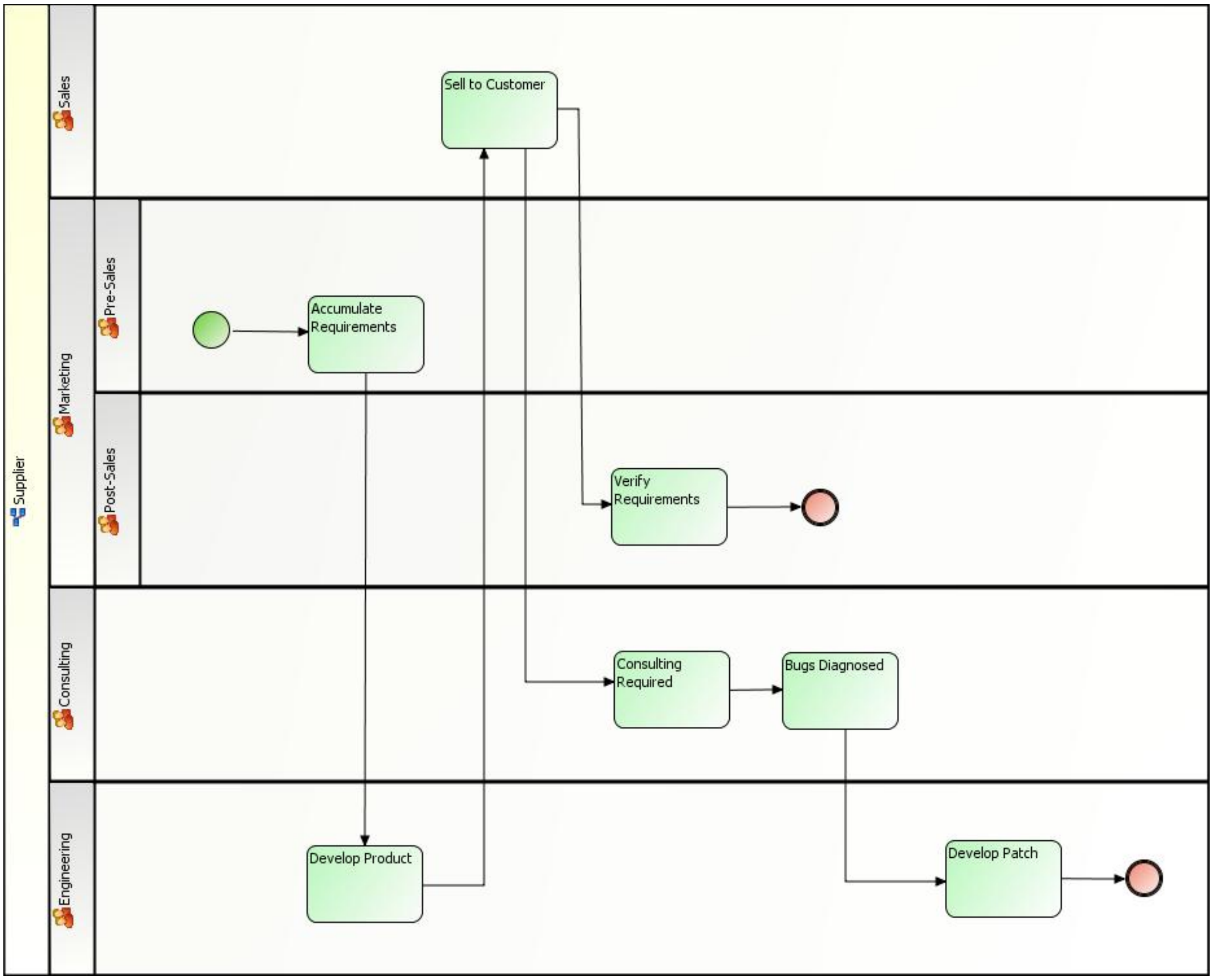
The lanes for already existing participant can be inserted by using the choice button from the top of the graph component. The lane will be inserted at the bottom of the Pool (or at the bottom of the already selected lane - this will be nested lane in that case).

Free text expression lane and **Common expression lane** are not representing participants but it is a special visualization of performer expressions for activities.

The Graph represents activity performers as swimlanes. When an activity is inserted or moved into a particular swimlane, it's performer will be updated to the participant represented by the swimlane. In the case of Common expression and Free text expression participants the activity's performer will be set to the expression defined as a property of the graph's common expression participant swimlane object. When the activity is inserted or moved into the free text expression participant any expression can be set for the activity performer field but by default performer expression won't be defined.

Together Workflow Editor allows you to **nest** lanes thus making possible to visually represent organizational structure. To nest one lane into another, select the parent lane and then create new lane either by selecting the participant it will represent using the lane choice button , or by selecting one of the lanes buttons (, ,) and pressing on the wanted parent lane. Here is a sample of the process with a Pool that contains nested lanes:

Figure 4.7. Nested lanes sample



Normal activity enables you to insert an activity that will be performed by a human (so called manual activity) - it will appear in the user's worklist.

Task-Applicationl activity enables the definition of applications that are required for the enactment engine to run in order to perform the activity.

Subflow activity is a type of activity whose implementation is another workflow process definition.

Block activity executes an ActivitySet (set of self-contained activities/transition maps). It is something like an embedded subflow process.

Route activity, in the forms of **Exclusive** and **Parallel** gateways, does not implement any action. It is used for synchronization and conditional branching only.

A **transition** binds two activities but it can also be a circular transition from a certain activity to itself. A straight line with an arrow pointing to the target object represents the transition in the Graph. Depending on the transition type, it will have a different color or additional graphical descriptions according to BPMN specification. The toolbar offers several kinds of transitions: the one describing uncontrolled flow →, the one describing conditional flow ⊸→(a circle on the left will be drawn according to BPMN spec only in the case the source of the transition is not a gateway), the one describing default flow (otherwise transition) and two kinds of exception transitions and (default exception). The type of a transition can be changed through its property panel.

There are two types of **artifacts** that can be inserted into the graph, data object artifact and text annotation . Artifacts describe e.g. document flow through the process, or make some comment that should be visible in the graph. The artifacts has to be connected to the activities using **associations**.

Associations are connecting an artifact with an activity. There are three types of associations offered by the TWE, directional , non-directional and bi-directional . The type of association can be changed through its property panel.

The Graph configuration is done through the *togwegraphcontroller.properties* file.

The following parameters can be set:

- Graph.FontSizedemo

  This property will set font size used in graph (for activity name, etc.). Values: integer

  NOTE: The graph font size can be set independently of the font size for the rest of the editor.

- Graph.GridSize

  The distance between position grid dots. Values: integer

- Graph.ShadowWidth

  The width of the shadow when displaying activities in the graph. Values: integer

- Graph.NameWrapping

  Defines if the text representing the names of activities and participants displayed in the graph will be wrapped if they are too long. Values: true/false

- Graph.ShowGrid

  Defines if the graph position grid will be shown. Values: true/false

- Graph.ShowIcon

  Defines if the graph elements should display their icons. Values: true/false

- Graph.ShowShadow

  Defines if activity shadows should be shown. Values: true/false

- Graph.ShowTransitionCondition

  Defines if transition conditions should be displayed in the graph. Values: true/false

- Graph.ShowTransitionNameForCondition

  Defines if the name of the transition should be shown instead of the condition of the transition. This setting only has effect if the Graph.ShowTransitionCondition parameter is set to true. Values: true/false

- Graph.DefaultTransitionStyle

Defines the default transition style when inserting new transitions in the graph. Possible values: NO_ROUTING_ORTHOGONAL, NO_ROUTING_SPLINE, NO_ROUTING_BEZIER, SIMPLE_ROUTING_ORTHOGONAL, SIMPLE_ROUTING_SPLINE and SIMPLE_ROUTING_BEZIER

- Graph.WrappingStyleWordStatus

  Defines if text wrapping should be based on whole words (if possible). This setting only has effect if the Graph.NameWrapping parameter is set to true. Values: true/false

- Graph.HistorySize

  Defines the size of the previous/next panel history. If the parameter is set to a value less than zero, the history is unlimited. Values: integer

- Graph.ActivityHeight

  The Graph's activity object height in pixels. Values: integer

- Graph.ActivityWidth

  The Graph's activity object width in pixels. Values: integer

- Graph.GatewayHeight

  The Graph's gateway activity (route activities) object height in pixels. Values: integer

- Graph.GatewayWidth

  The Graph's gateway activity (route activities) object width in pixels. Values: integer

- Graph.EventRadius

  The Graph's start/end event activity object radius in pixels. Values: integer

- Graph.DataObjectHeight

  The Graph's "dataobject" artifact object height in pixels. Values: integer

- Graph.DataObjectWidth

  The Graph's "dataobject" artifact object width in pixels. Values: integer

- Graph.LaneMinHeight

  The lane's graph object minimum height in pixels. Values: integer

- Graph.LaneMinWidth

  The lane's graph object minimum width in pixels. Values: integer

- Graph.LaneNameWidth

  Defines how much space (in pixels) of the swimlane should be reserved to display name. Values: integer

- Graph.ActivitySelectedColor

  The color for selected activities. Values: R=?,G=?,B=? (where ? represents an integer from 0 to 255).

- Graph.BackgroundColor

The Graph background color. Values: R=?,G=?,B=? (where ? represents an integer from 0 to 255).

- Graph.HandleColor

  The Color of break/routepoints of transitions (they are only visible when a transition is selected). Values: R=?,G=?,B=? (where ? represents an integer from 0 to 255).

- Graph.GridColor

  The color for position grid dots. Values: R=?,G=?,B=? (where ? represents an integer from 0 to 255).

- Graph.MarqueeColor

  The color for marquee rectangles. Values: R=?,G=?,B=? (where ? represents an integer from 0 to 255).

- Graph.ParticipantBorderColor

  The color for participant's swimlane borders. Values: R=?,G=?,B=? (where ? represents an integer from 0 to 255).

- Graph.LaneCommonExpressionColor

  The color for the common expression lane. Values: R=?,G=?,B=? (where ? represents an integer from 0 to 255).

- Graph.LaneFreeTextExpressionColor

  The color for the free text expression lane. Values: R=?,G=?,B=? (where ? represents an integer from 0 to 255).

- Graph.TextColor

  The color of the Graph text. Values: R=?,G=?,B=? (where ? represents an integer from 0 to 255).

- Graph.StartEventColor

  Start event activities color. Values: R=?,G=?,B=? (where ? represents an integer from 0 to 255).

- Graph.EndEventColor

  End event activities color. Values: R=?,G=?,B=? (where ? represents an integer from 0 to 255).

As with all other components, you are also able to change the content of the toolbar and the order of toolbar buttons.

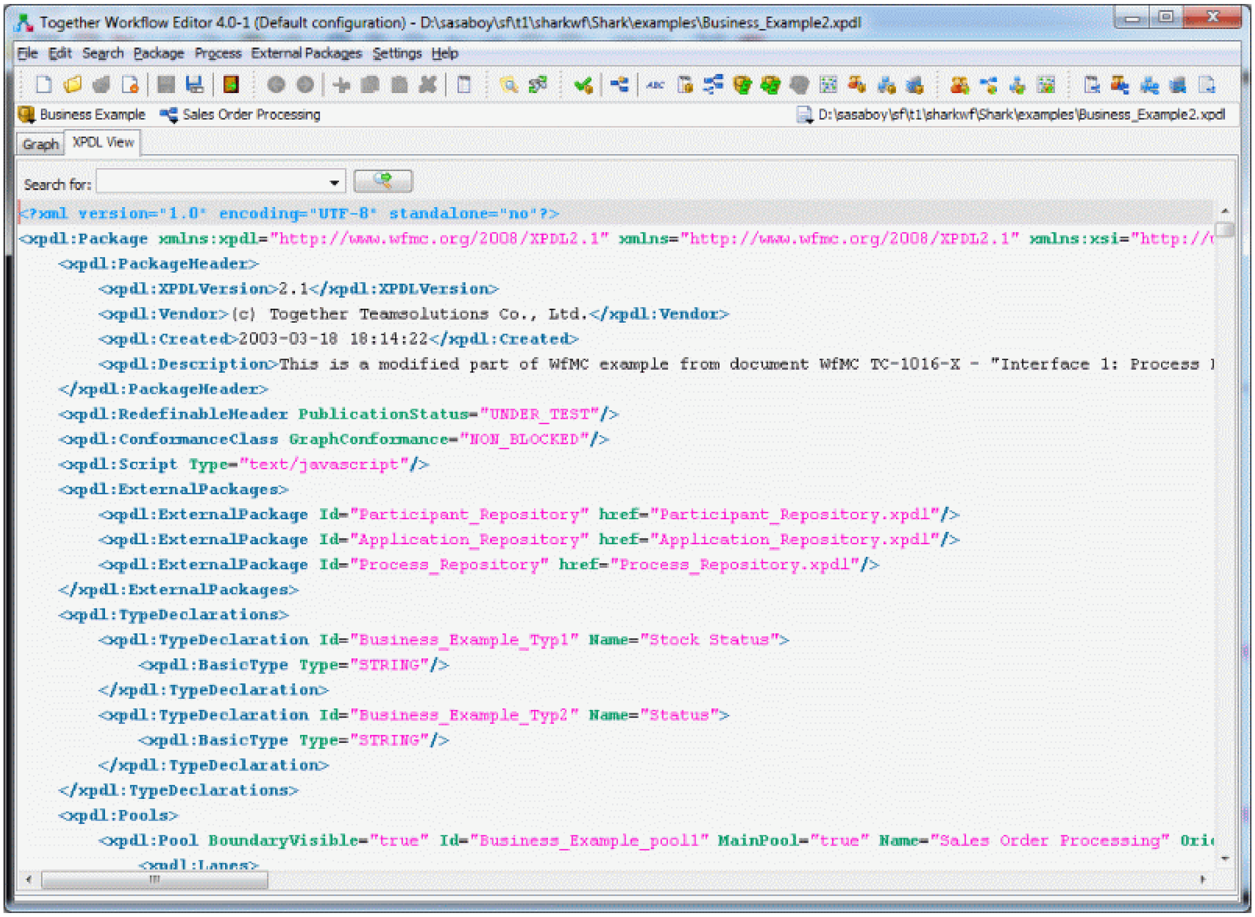
XPDL View

The XPDL view provides an XPDL text view of the selected elements. It shows the structure of the selected XML element as it will be saved in the XPDL file. Every time the selection is changed or a selected element is being updated, the XPDL View is automatically updated also.

If the selected element is a whole XPDL package then the XPDL View displays the whole package definition in the exact form as it will be written into the XPDL file (when the document is saved). Otherwise the XPDL View will display the part of the XPDL which holds the definition of the selected element. So, when a particular process is selected, the XPDL view displays the process definition part of XPDL as it will be written to the XPDL file, etc.

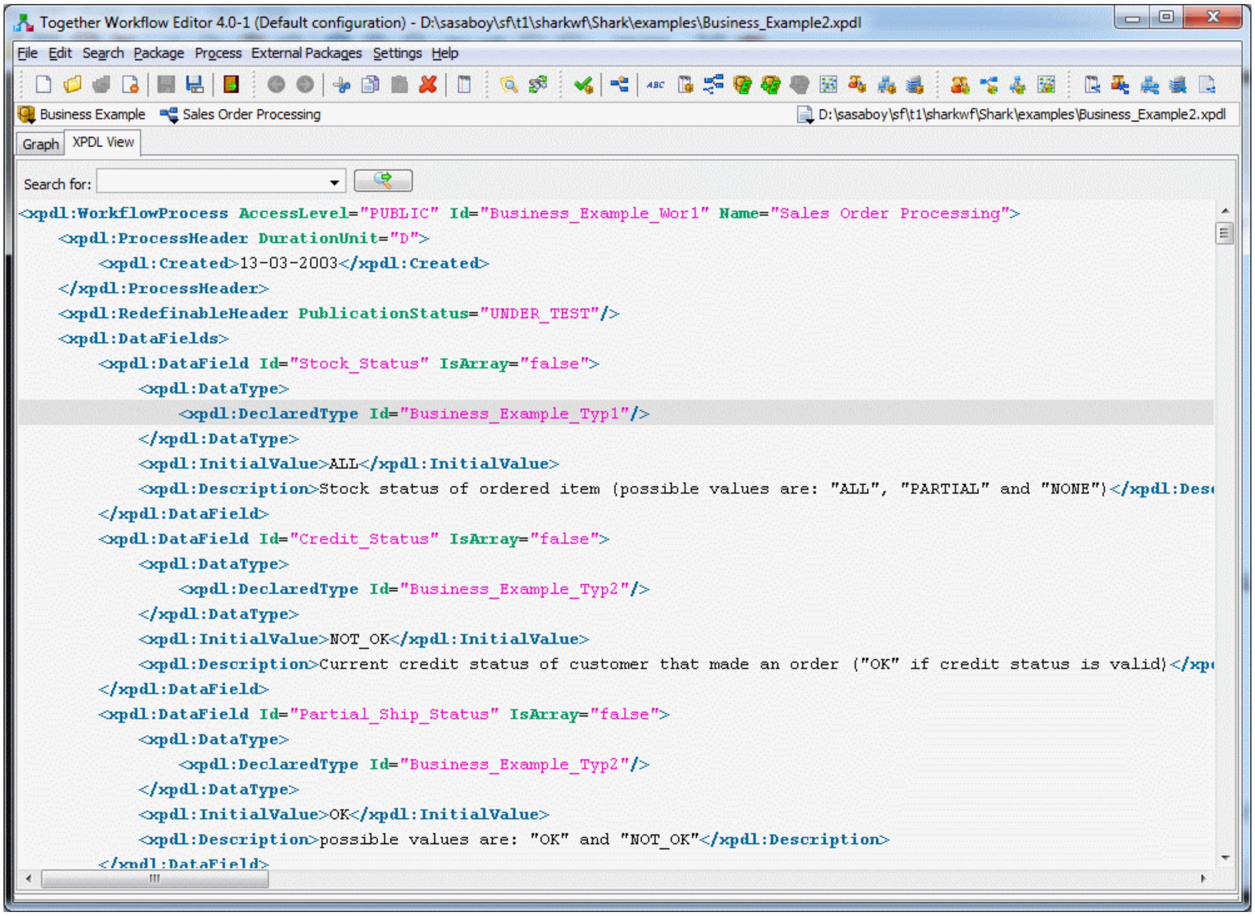
The following examples show different XPDL Views depending on the element type that is currently selected:

Figure 4.8. XPDL View of a Package


Together Workflow Editor 4.0-1 (Default configuration) - D:\sasaboy\sf\t1\sharkwf\Shark\examples\Business_Example2.xpdl
File Edit Search Package Process External Packages Settings Help
Business Example
Sales Order Processing
D:\sasaboy\sf\t1\sharkwf\Shark\examples\Business_Example2.xpdl
Graph
XPDL View
Search for:

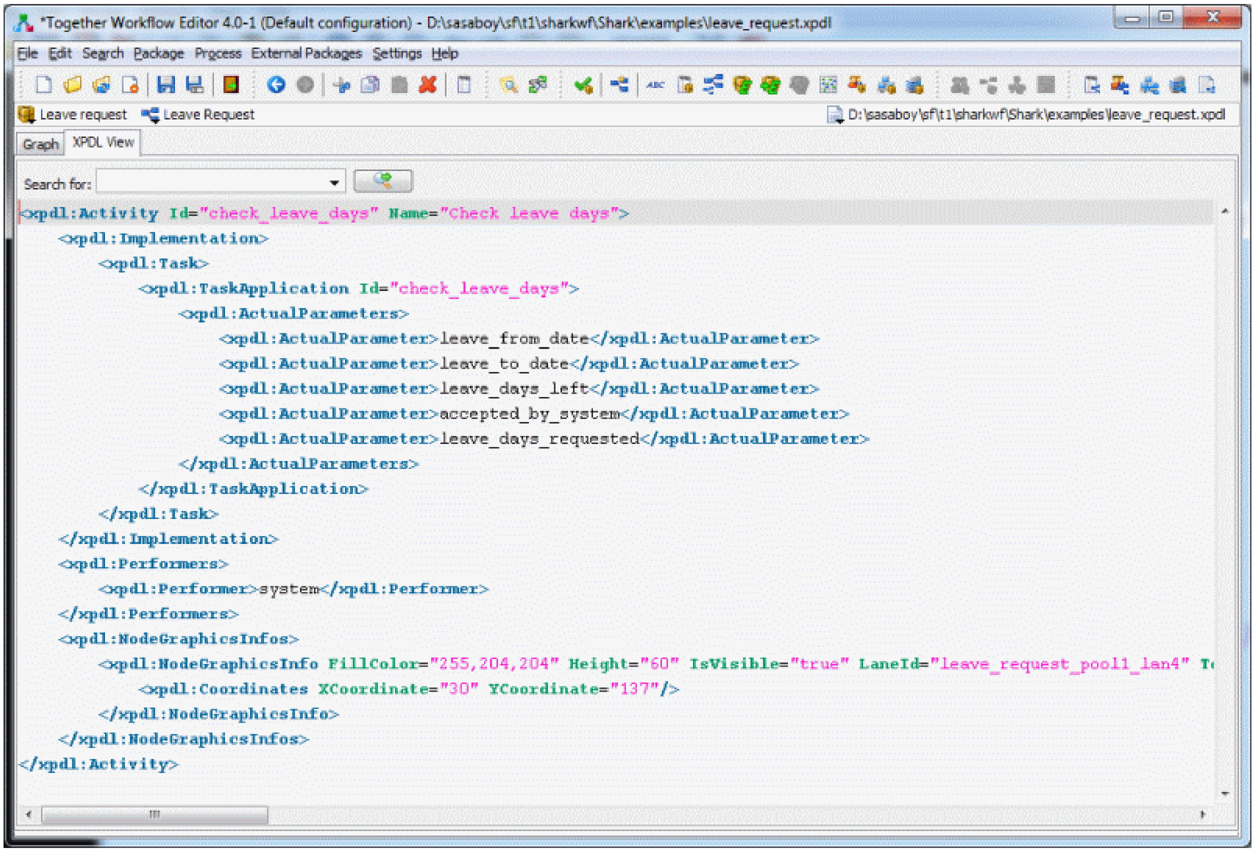
```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<xpdl:Package xmlns:xpdl="http://www.wfmc.org/2008/XPDL2.1" xmlns="http://www.wfmc.org/2008/XPDL2.1" xmlns:xsi="http://
    <xpdl:PackageHeader>
        <xpdl:XPDLVersion>2.1</xpdl:XPDLVersion>
        <xpdl:Vendor>(c) Together Teamsolutions Co., Ltd.</xpdl:Vendor>
        <xpdl:Created>2003-03-18 18:14:22</xpdl:Created>
        <xpdl:Description>This is a modified part of WfMC example from document WfMC TC-1016-X - "Interface 1: Process
    </xpdl:PackageHeader>
    <xpdl:RedefinableHeader PublicationStatus="UNDER_TEST"/>
    <xpdl:ConformanceClass GraphConformance="NON_BLOCKED"/>
    <xpdl:Script Type="text/javascript"/>
    <xpdl:ExternalPackages>
        <xpdl:ExternalPackage Id="Participant_Repository" href="Participant_Repository.xpdl"/>
        <xpdl:ExternalPackage Id="Application_Repository" href="Application_Repository.xpdl"/>
        <xpdl:ExternalPackage Id="Process_Repository" href="Process_Repository.xpdl"/>
    </xpdl:ExternalPackages>
    <xpdl:TypeDeclarations>
        <xpdl:TypeDeclaration Id="Business_Example_Typ1" Name="Stock Status">
            <xpdl:BasicType Type="STRING"/>
        </xpdl:TypeDeclaration>
        <xpdl:TypeDeclaration Id="Business_Example_Typ2" Name="Status">
            <xpdl:BasicType Type="STRING"/>
        </xpdl:TypeDeclaration>
    </xpdl:TypeDeclarations>
    <xpdl:Pools>
        <xpdl:Pool BoundaryVisible="true" Id="Business_Example_pool1" MainPool="true" Name="Sales Order Processing" Ori
            <xpdl:Lanes>
```

Figure 4.9. XPDL View of a Process


Together Workflow Editor 4.0-1 (Default configuration) - D:\sasaboy\sf\t1\sharkwf\Shark\examples\Business_Example2.xpdl
File Edit Search Package Process External Packages Settings Help
Business Example
Sales Order Processing
D:\sasaboy\sf\t1\sharkwf\Shark\examples\Business_Example2.xpdl
Graph
XPDL View
Search for:

```xml
<xpdl:WorkflowProcess AccessLevel="PUBLIC" Id="Business_Example_Wor1" Name="Sales Order Processing">
    <xpdl:ProcessHeader DurationUnit="D">
        <xpdl:Created>13-03-2003</xpdl:Created>
    </xpdl:ProcessHeader>
    <xpdl:RedefinableHeader PublicationStatus="UNDER_TEST"/>
    <xpdl:DataFields>
        <xpdl:DataField Id="Stock_Status" IsArray="false">
            <xpdl:DataType>
                <xpdl:DeclaredType Id="Business_Example_Typ1"/>
            </xpdl:DataType>
            <xpdl:InitialValue>ALL</xpdl:InitialValue>
            <xpdl:Description>Stock status of ordered item (possible values are: "ALL", "PARTIAL" and "NONE")</xpdl:Des
        </xpdl:DataField>
        <xpdl:DataField Id="Credit_Status" IsArray="false">
            <xpdl:DataType>
                <xpdl:DeclaredType Id="Business_Example_Typ2"/>
            </xpdl:DataType>
            <xpdl:InitialValue>NOT_OK</xpdl:InitialValue>
            <xpdl:Description>Current credit status of customer that made an order ("OK" if credit status is valid)</xp
        </xpdl:DataField>
        <xpdl:DataField Id="Partial_Ship_Status" IsArray="false">
            <xpdl:DataType>
                <xpdl:DeclaredType Id="Business_Example_Typ2"/>
            </xpdl:DataType>
            <xpdl:InitialValue>OK</xpdl:InitialValue>
            <xpdl:Description>possible values are: "OK" and "NOT_OK"</xpdl:Description>
        </xpdl:DataField>
```

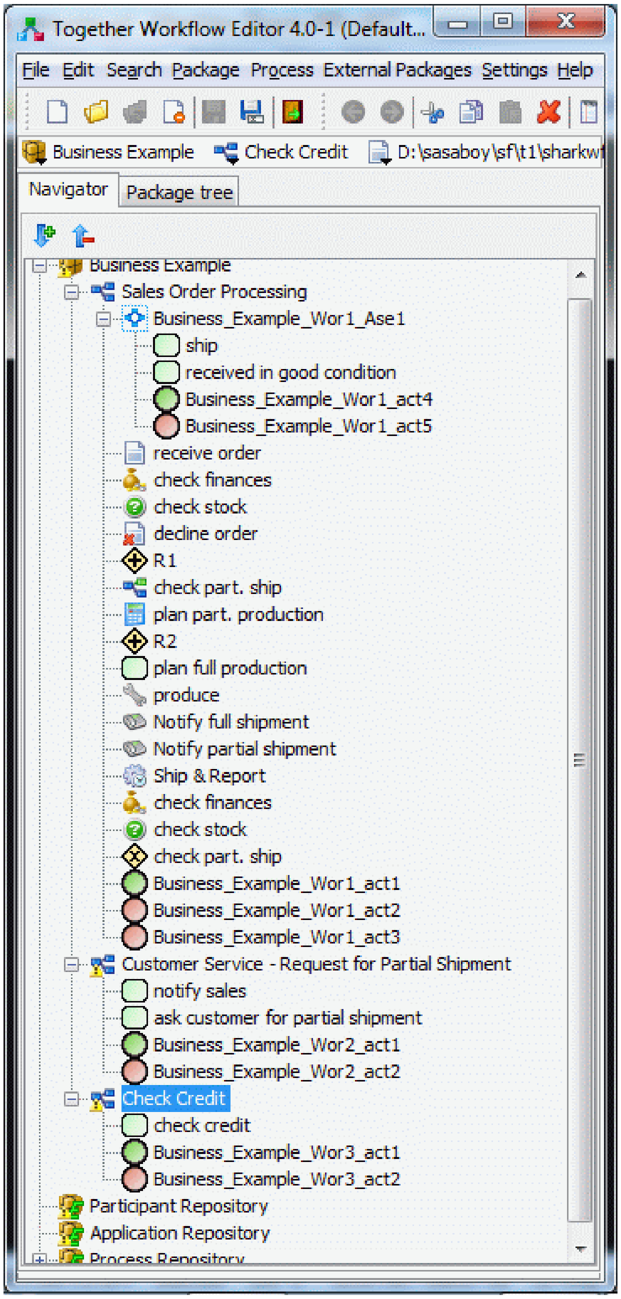
Figure 4.10. XPDL View of an Activity


*Together Workflow Editor 4.0-1 (Default configuration) - D:\sasaboy\sf\t1\sharkwf\Shark\examples\leave_request.xpdl
File Edit Search Package Process External Packages Settings Help
Leave request
Leave Request
D:\sasaboy\sf\t1\sharkwf\Shark\examples\leave_request.xpdl
Graph
XPDL View
Search for:

```
<xpdl:Activity Id="check_leave_days" Name="Check leave days">
    <xpdl:Implementation>
        <xpdl:Task>
            <xpdl:TaskApplication Id="check_leave_days">
                <xpdl:ActualParameters>
                    <xpdl:ActualParameter>leave_from_date</xpdl:ActualParameter>
                    <xpdl:ActualParameter>leave_to_date</xpdl:ActualParameter>
                    <xpdl:ActualParameter>leave_days_left</xpdl:ActualParameter>
                    <xpdl:ActualParameter>accepted_by_system</xpdl:ActualParameter>
                    <xpdl:ActualParameter>leave_days_requested</xpdl:ActualParameter>
                </xpdl:ActualParameters>
            </xpdl:TaskApplication>
        </xpdl:Task>
    </xpdl:Implementation>
    <xpdl:Performers>
        <xpdl:Performer>system</xpdl:Performer>
    </xpdl:Performers>
    <xpdl:NodeGraphicsInfos>
        <xpdl:NodeGraphicsInfo FillColor="255,204,204" Height="60" IsVisible="true" LaneId="leave_request_pool1_lan4" T
            <xpdl:Coordinates XCoordinate="30" YCoordinate="137"/>
        </xpdl:NodeGraphicsInfo>
    </xpdl:NodeGraphicsInfos>
</xpdl:Activity>
```

The XPDL view also offers a search box for finding desired text fragments in the currently shown XPDL fragment.

Navigator


Together Workflow Editor 4.0-1 (Default...
File Edit Search Package Process External Packages Settings Help
Business Example
Check Credit
D:\sasaboy\sf\t1\sharkw
Navigator
Package tree
Business Example
Sales Order Processing
Business_Example_Wor1_Ase1
ship
received in good condition
Business_Example_Wor1_act4
Business_Example_Wor1_act5
receive order
check finances
check stock
decline order
R1
check part. ship
plan part. production
R2
plan full production
produce
Notify full shipment
Notify partial shipment
Ship & Report
check finances
check stock
check part. ship
Business_Example_Wor1_act1
Business_Example_Wor1_act2
Business_Example_Wor1_act3
Customer Service - Request for Partial Shipment
notify sales
ask customer for partial shipment
Business_Example_Wor2_act1
Business_Example_Wor2_act2
Check Credit
check credit
Business_Example_Wor3_act1
Business_Example_Wor3_act2
Participant Repository
Application Repository
Process Repository

The Navigator displays a hierarchical view of the XPDL model. It shows packages, processes, activity sets and activities. It is also very useful for finding XPDL elements which are not valid according to the XPDL specification. These elements will be marked with error (✖) or warning (⚠) icons.

The Navigator makes navigation through the XPDL model easier and offers a good way of element selection. It can also ease editing of XPDL by opening property dialogs directly from the elements in the tree.

The available actions in the Navigator toolbar are *expand all* and *collapse all*. When expand all is selected the hierarchical tree will be expanded so that every element is visible. When collapse all is performed, only top element(s) will be made visible.

A right mouse click on tree will display a popup menu. Besides the standard edit functions (cut, copy, paste, delete and edit properties) there are Expand all, Collapse all, Duplicate and References.

Expand all and Collapse all will act like the actions on the toolbar with the selected element used as the root element of the action. So Expand all will make all children of the selected element visible.

Duplicate will create a copy of the selected element. The only difference will be in the element id and name attributes (if they exist) which will get new values.

The References action will search for references to the selected element in the whole XPDL model.

The background and the selection colors can be set through the packagenavigator.properties file.

BackgroundColor. Values: R=?,G=?,B=? (where ? represents an integer from 0 to 255).

and

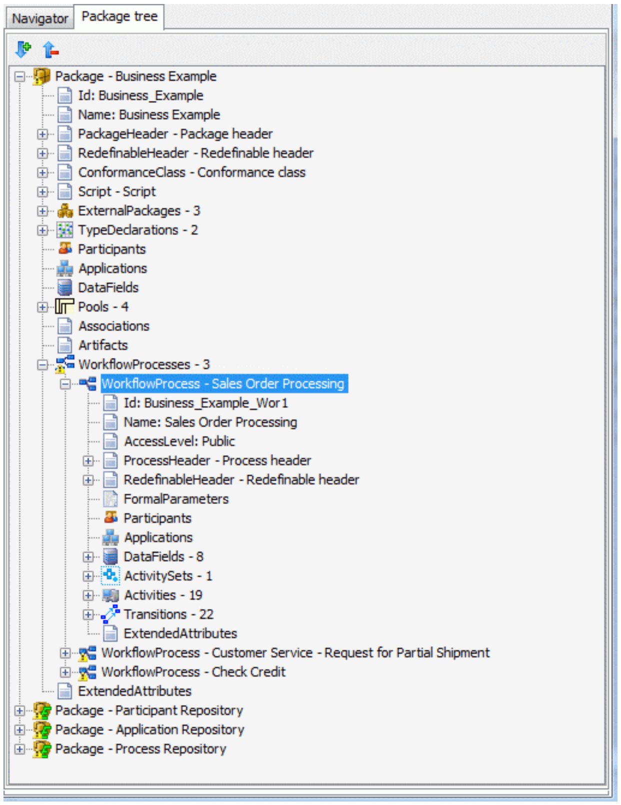
SelectionColor. Values: R=?,G=?,B=? (where ? represents an integer from 0 to 255).

As with all other components you are also able to change the content of the toolbar and the order of toolbar buttons.

Package tree

Similar to the Navigator the Package tree also displays a hierarchical view of the XPDL model. The difference between these two trees is the level of details. While the Navigator just shows packages, processes, activity sets and activities, The Package tree shows almost every element of the XPDL model.

The Package tree offers the most detailled hierarchical view of the XPDL model. It graphically displays the whole hierarchy tree of all XPDL elements of the model you are editing, Id attributes, names, the package header, etc.


Navigator
Package tree
Package - Business Example
Id: Business_Example
Name: Business Example
PackageHeader - Package header
RedefinableHeader - Redefinable header
ConformanceClass - Conformance class
Script - Script
ExternalPackages - 3
TypeDeclarations - 2
Participants
Applications
DataFields
Pools - 4
Associations
Artifacts
WorkflowProcesses - 3
WorkflowProcess - Sales Order Processing
Id: Business_Example_Wor1
Name: Sales Order Processing
AccessLevel: Public
ProcessHeader - Process header
RedefinableHeader - Redefinable header
FormalParameters
Participants
Applications
DataFields - 8
ActivitySets - 1
Activities - 19
Transitions - 22
ExtendedAttributes
WorkflowProcess - Customer Service - Request for Partial Shipment
WorkflowProcess - Check Credit
ExtendedAttributes
Package - Participant Repository
Package - Application Repository
Package - Process Repository

The Package tree makes navigation through the XPDL elements easierIt can also ease editing of XPDL by opening property dialogs directly from the elements in the tree.

The available actions in the Package tree toolbar are *expand all* and *collapse all*. When expand all is selected the hierarchical tree will be expanded so that every element is visible. When collapse all is performed, only top element(s) will be made visible.

A right mouse click on tree will display a popup menu. Besides the standard edit functions (cut, copy, paste, delete and edit properties) there are Expand all, Collapse all, Duplicate and References.

Expand all and Collapse all will act like the actions on the toolbar with the selected element used as the root element of the action. So Expand all will make all children of the selected element visible.

Duplicate will create a copy of the selected element. The only difference will be in the element id and name attributes (if they exist) which will get new values.

The References action will search for references to the selected element in the whole XPDL model.

The Package tree component can be configured through detailpackagenavigator.properties file.

- *BackgroundColor* - background color
- *SelectionColor* - selection color

You are able to customize the Package tree in order to hide some complex element's sub-elements. For example: To hide Activity's Id, Deadlines, Priority and Limit, you should set the property

HideSubElements.Activity

to the following value:

```
HideSubElements.Activity = Id Deadlines Priority Limit
```

You are also able to customize which elements of some collection shouldn't be displayed. For example: To hide extended attributes which name attribute is "SpecEA" or "EASpec", you can define the property:

```
HideElements.ExtendedAttributes.Name = SpecEA EASpec
```

If you want to hide all elements use *. For example: To hide all extended attributes use:

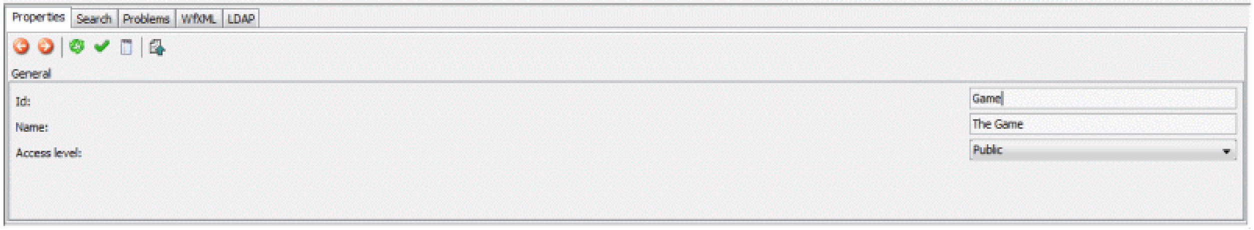
HideElements.ExtendedAttributes.Name = *

As with all other components you are also able to change the content of the toolbar and the order of toolbar buttons.

Properties component

Properties component displays selected XPDL element's properties.

Figure 4.11. Properties component



Id: Game
Name: The Game
Access level: Public

By selecting an element, this component automatically displays the most relevant data about that element (such as name, id etc.).

Besides data, this component also offers a couple of operations:

←	Displays properties of previously selected element (that element again becomes selected).
→	Displays properties of element selected after this element selection (the following element again becomes selected).
⟲	Discards new values of properties (reverts to the values applied with last 'apply' action call).
✓	Applies new values of properties.
☐	Opens a dialog with a full set of the properties for the displayed element.
⇧	Shows parent XPDL element panel.

This component can be configured through propertiespanelcomponent.properties file.

- *InlinePanel.ShowModifiedWarning* - if set to true, and if user wants to leave the property panel (by using show previous, show next, show parent element panel actions or actions for closing the dialog) of the element for which he previously changed some properties (within the panel), the user is asked if he wants to save the changes, leave the panel without saving changes or to cancel the action.
- *InlinePanel.DisplayTitle* - if set to true, panel will display element name beneath toolbar.
- *HistorySize* - defines the size of the previous/next panel history. If set to the value less than zero, the history is unlimited.

- *Toolbar.ActionOrder.defaultToolbar* - defines the content and the order of the dialog's toolbar.

There are properties which can be adjusted to fine-tune the basic element property panels L&F, such as alignment, TOP, BOTTOM, RIGHT, LEFT empty space in the panels, the width and height of the text boxes, etc. These are defined by the following set of the properties:

```
XMLBasicPanel.RightAllignment=false
XMLBasicPanel.EmptyBorder.TOP=0
XMLBasicPanel.EmptyBorder.LEFT=3
XMLBasicPanel.EmptyBorder.BOTTOM=4
XMLBasicPanel.EmptyBorder.RIGHT=3
XMLBasicPanel.SimplePanelTextWidth=250
XMLBasicPanel.SimplePanelTextHeight=20
```

You are able to customize so called "group" panels, used to display some complex elements, in order to hide some complex element sub-elements. For example, in order not to display Activity's Id, Deadlines, Priority and Limit, you should set the property

HideSubElements.XMLGroupPanel.Activity

to the following value:

```
HideSubElements.XMLGroupPanel.Activity = Id Deadlines Priority Limit
```

You are able to customize which elements of some collection shouldn't be displayed within so called "table" panels. For example, if you don't want to display extended attributes which name attribute is "SpecEA" or "EASpec", you can define the property:

```
HideElements.XMLTablePanel.ExtendedAttributes.Name = SpecEA EASpec
```

You are able to customize so called "table" panels, used to display some complex element collections, in order to specify which sub-elements will be shown as a table columns. For example, when displaying activities, you can specify to show Activity's Id, Name, Performer, Type, Start mode, Finish mode and Deadlines:

ShowColumns.XMLTablePanel.Activities = Id Name Performer Type StartMode FinishMode Deadlines

There is another customization possible for the so called "combo box" panel. Hence, you can define for which elements the combo box will be disabled (by default nothing is disabled). For example, if you want to disable combo boxes for displaying Activity's Performer and Transition's From and To properties, you should specify the following:

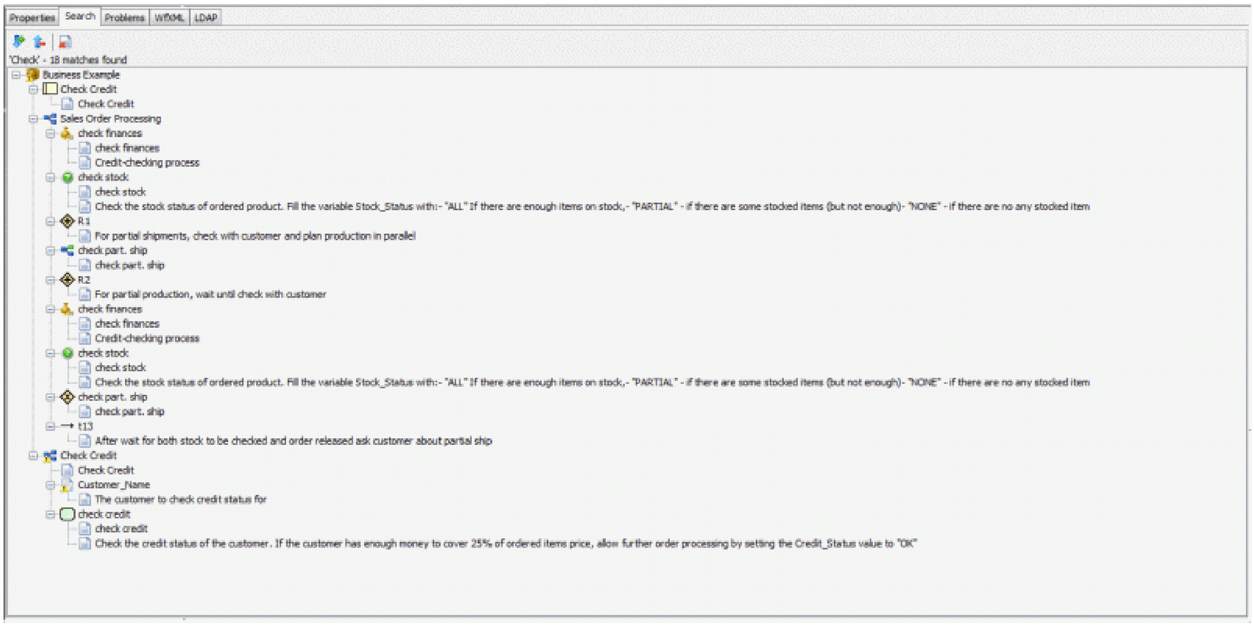
```
XMLComboPanel.DisableCombo = Performer From To
```

As with all other components, you are also able to change the content of the toolbar, and the order of toolbar buttons.

Search component

Search component displays results of a *References* or *Search* action. It shows all elements that have a reference to the selected element in the first case, or all the elements found based on entered criteria in second case. Results are shown in hierarchical view.

Figure 4.12. Search Results



```
'Check' - 18 matches found
Business Example
    Check Credit
        Check Credit
    Sales Order Processing
        check finances
            check finances
            Credit-checking process
        check stock
            check stock
            Check the stock status of ordered product. Fill the variable Stock_Status with:- "ALL" If there are enough items on stock,- "PARTIAL" - if there are some stocked items (but not enough)- "NONE" - if there are no any stocked item
        R1
            For partial shipments, check with customer and plan production in parallel
        check part. ship
            check part. ship
        R2
            For partial production, wait until check with customer
        check finances
            check finances
            Credit-checking process
        check stock
            check stock
            Check the stock status of ordered product. Fill the variable Stock_Status with:- "ALL" If there are enough items on stock,- "PARTIAL" - if there are some stocked items (but not enough)- "NONE" - if there are no any stocked item
        check part. ship
            check part. ship
        t13
            After wait for both stock to be checked and order released ask customer about partial ship
    Check Credit
        Check Credit
        Customer_Name
            The customer to check credit status for
        check credit
            check credit
            Check the credit status of the customer. If the customer has enough money to cover 25% of ordered items price, allow further order processing by setting the Credit_Status value to "OK"
```

Besides search results, this component also offers a couple of operations:

⬇	Expands hierarchical results view.
⬆	Collapses hierarchical results view.
✖	Clears the result page.

While left click on result select that element, right click will also bring popup menu. Popup menu has action for expanding and collapsing, bringing up properties and performing *References* action for selected element.

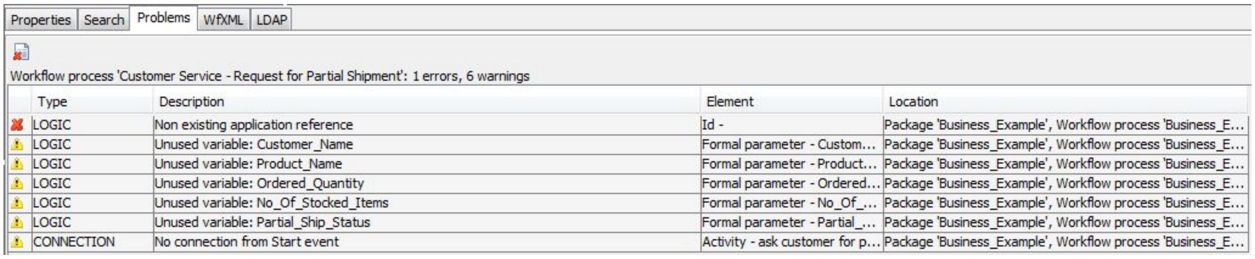
The background color could be changed through searchnavigator.properties file. (BackgroundColor = R=?,G=?,B=?)

As with all other components, you are also able to change the content of the toolbar, and the order of toolbar buttons.

Problems component

Problems component lists all the problems currently present in the XPDL model(s). If option for design-time validation (found in togwecontroller.property) is set to 'true', this list will be automatically refreshed (except for schema errors).

Figure 4.13. Problems component



Workflow process 'Customer Service - Request for Partial Shipment': 1 errors, 6 warnings

Type	Description	Element	Location
✖ LOGIC	Non existing application reference	Id -	Package 'Business_Example', Workflow process 'Business_E...
⚠ LOGIC	Unused variable: Customer_Name	Formal parameter - Custom...	Package 'Business_Example', Workflow process 'Business_E...
⚠ LOGIC	Unused variable: Product_Name	Formal parameter - Product...	Package 'Business_Example', Workflow process 'Business_E...
⚠ LOGIC	Unused variable: Ordered_Quantity	Formal parameter - Ordered...	Package 'Business_Example', Workflow process 'Business_E...
⚠ LOGIC	Unused variable: No_Of_Stocked_Items	Formal parameter - No_Of_...	Package 'Business_Example', Workflow process 'Business_E...
⚠ LOGIC	Unused variable: Partial_Ship_Status	Formal parameter - Partial_...	Package 'Business_Example', Workflow process 'Business_E...
⚠ CONNECTION	No connection from Start event	Activity - ask customer for p...	Package 'Business_Example', Workflow process 'Business_E...

Results of action 'check validity' are also shown here, so if design-time validation is off, you can always re-check the XPDL model validity by calling this action from the Package's menu/toolbar.

The result table has five columns:

- The first column (without title) is representing a type of a problem, which can be either error or warning and it's marked with appropriate icon.
- The second column specifies the type of the error/warning (connection, logic, conformance or schema)
- The third column is a description of the problem.
- The fourth column displays the exact element where the problem occurs
- The fifth column is a location of the first relevant parent of the problematic element (it could also be the problematic element itself).

Clicking on Element column selects the exact element which has an error/warning described. By clicking on any other column, appropriate parent element will be selected (e.g. if error is with actual parameter, clicking on element will select actual parameter, while clicking on some other column will select activity) - as already explained, sometimes these two are the same.

On the other hand, double-clicking opens a dialog for editing problematic element.

There is also a possibility to sort entries by the column - just click to the column header, and entries will be sorted by this column.

Problem component background color could be changed through problemnavigator.properties file. (BackgroundColor = R=?,G=?,B=?)

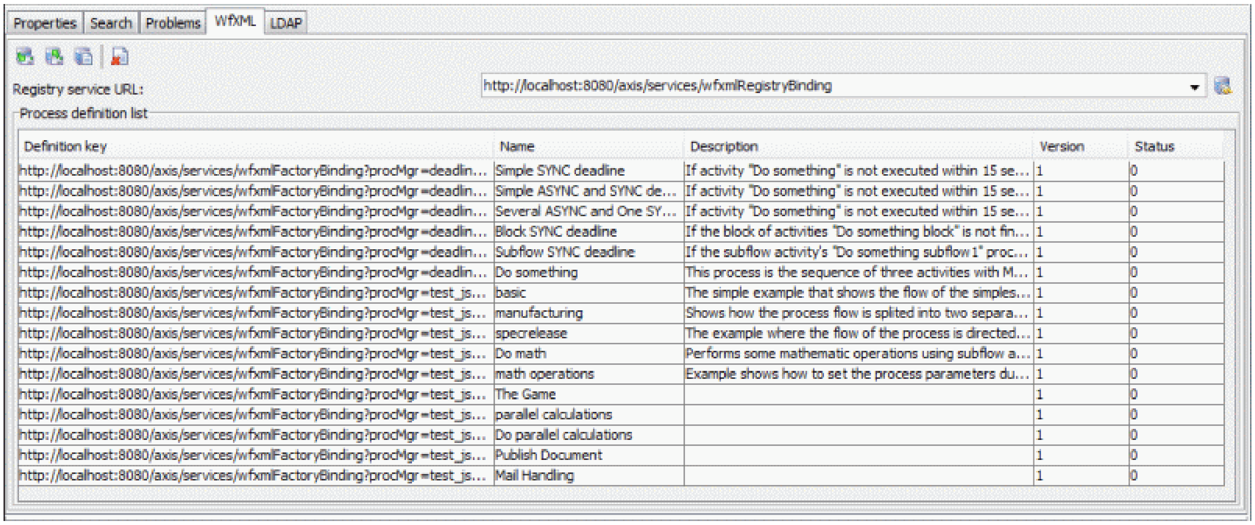
As with all other components, you are also able to change the content of the toolbar, and the order of toolbar buttons.

WfXML component

Having this component, you are able to connect to a workflow engine which supports WfMC interface defined by WfMC, and to manage XPDL files on the engine.

To connect to a workflow engine, valid registry service URL must be entered. After you enter it, and press the button right to the URL, you should connect to the engine, and get a list of the process definitions already uploaded to the engine, as on the picture below:

Figure 4.14. WfXML Component



Registry service URL: http://localhost:8080/axis/services/wfxmlRegistryBinding

Definition key	Name	Description	Version	Status
http://localhost:8080/axis/services/wfxmlFactoryBinding?procMgr=deadlin…	Simple SYNC deadline	If activity "Do something" is not executed within 15 se…	1	0
http://localhost:8080/axis/services/wfxmlFactoryBinding?procMgr=deadlin…	Simple ASYNC and SYNC de…	If activity "Do something" is not executed within 15 se…	1	0
http://localhost:8080/axis/services/wfxmlFactoryBinding?procMgr=deadlin…	Several ASYNC and One SY…	If activity "Do something" is not executed within 15 se…	1	0
http://localhost:8080/axis/services/wfxmlFactoryBinding?procMgr=deadlin…	Block SYNC deadline	If the block of activities "Do something block" is not fin…	1	0
http://localhost:8080/axis/services/wfxmlFactoryBinding?procMgr=deadlin…	Subflow SYNC deadline	If the subflow activity's "Do something subflow 1" proc…	1	0
http://localhost:8080/axis/services/wfxmlFactoryBinding?procMgr=deadlin…	Do something	This process is the sequence of three activities with M…	1	0
http://localhost:8080/axis/services/wfxmlFactoryBinding?procMgr=test_js…	basic	The simple example that shows the flow of the simples…	1	0
http://localhost:8080/axis/services/wfxmlFactoryBinding?procMgr=test_js…	manufacturing	Shows how the process flow is splited into two separa…	1	0
http://localhost:8080/axis/services/wfxmlFactoryBinding?procMgr=test_js…	specrelease	The example where the flow of the process is directed…	1	0
http://localhost:8080/axis/services/wfxmlFactoryBinding?procMgr=test_js…	Do math	Performs some mathematic operations using subflow a…	1	0
http://localhost:8080/axis/services/wfxmlFactoryBinding?procMgr=test_js…	math operations	Example shows how to set the process parameters du…	1	0
http://localhost:8080/axis/services/wfxmlFactoryBinding?procMgr=test_js…	The Game		1	0
http://localhost:8080/axis/services/wfxmlFactoryBinding?procMgr=test_js…	parallel calculations		1	0
http://localhost:8080/axis/services/wfxmlFactoryBinding?procMgr=test_js…	Do parallel calculations		1	0
http://localhost:8080/axis/services/wfxmlFactoryBinding?procMgr=test_js…	Publish Document		1	0
http://localhost:8080/axis/services/wfxmlFactoryBinding?procMgr=test_js…	Mail Handling		1	0

The toolbar for this component contains buttons for performing actions for managing XPDLs on the engine. The actions are described in the following table:

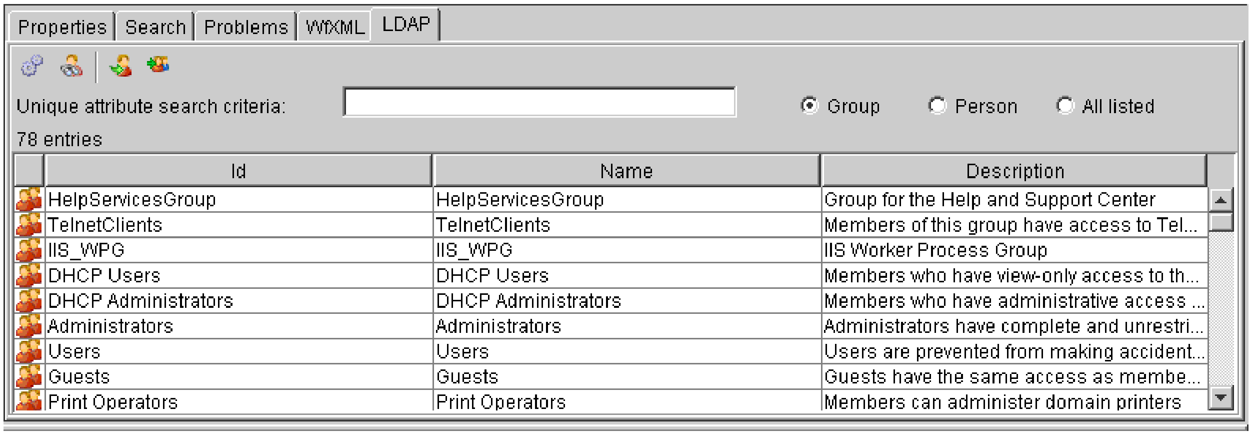
	Downloads XPDL from the engine.
	Uploads XPDL to the engine.
	Updates XPDL to the engine.
	Clears the history (which you are able to select from a combo box) of most recently entered URLs of WfXML compliant engines.

As with all other components, you are also able to change the content of the toolbar, and the order of toolbar buttons.

LDAP component

This component enables you to connect to the LDAP based system, search for users and groups and import them into XPDL as Participants.

Figure 4.15. LDAP Component



Unique attribute search criteria:

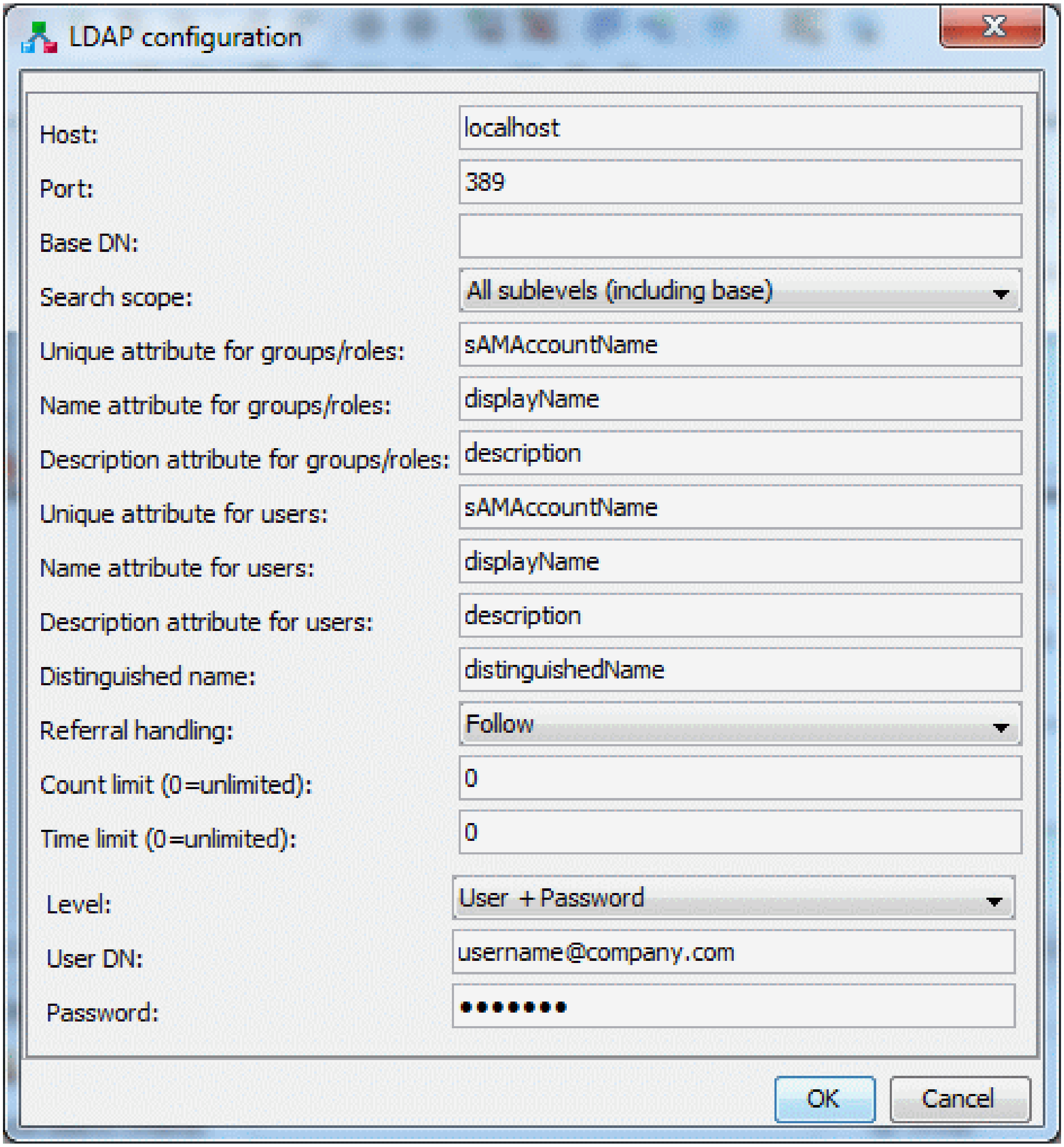
● Group ○ Person ○ All listed

78 entries

Id	Name	Description
HelpServicesGroup	HelpServicesGroup	Group for the Help and Support Center
TelnetClients	TelnetClients	Members of this group have access to Tel...
IIS_WPG	IIS_WPG	IIS Worker Process Group
DHCP Users	DHCP Users	Members who have view-only access to th...
DHCP Administrators	DHCP Administrators	Members who have administrative access ...
Administrators	Administrators	Administrators have complete and unrestri...
Users	Users	Users are prevented from making accident...
Guests	Guests	Guests have the same access as membe...
Print Operators	Print Operators	Members can administer domain printers

To be able to do this, you first need to configure LDAP parameters by selecting corresponding action from component's toolbar. This will bring you a dialog where you define how to access LDAP, names of attributes representing group and users, etc. as shown on the picture below:

Figure 4.16. LDAP Configuration Dialog



Host: localhost
Port: 389
Base DN:
Search scope: All sublevels (including base)
Unique attribute for groups/roles: sAMAccountName
Name attribute for groups/roles: displayName
Description attribute for groups/roles: description
Unique attribute for users: sAMAccountName
Name attribute for users: displayName
Description attribute for users: description
Distinguished name: distinguishedName
Referral handling: Follow
Count limit (0=unlimited): 0
Time limit (0=unlimited): 0
Level: User + Password
User DN: username@company.com
Password: •••••••

The toolbar for this component contains buttons for performing actions for managing LDAP access. The actions are described in the following table:

⚙	Opens a dialog to configure parameters neccessary to connect to an LDAP and search for users and groups.
🔍	Performs a LDAP search based on parameters entered in as a search criteria, and displays results in a table.
👤	Imports selected LDAP entries as XPDL Participants.

	Imports all LDAP entries as XPDL Participants.

As with all other components, you are also able to change the content of the toolbar, and the order of toolbar buttons.

The advantage of using "LDAP defined participants" is that one can connect the LDAP server of its organization and create processes using the real user/group logins for participant Ids. When the engine enacts such a process, it could also browse the LDAP server for the certain users/groups, so you do not have to do "participant to real user mapping".

The default LDAP properties can be defined in ldap.properties configuration file:

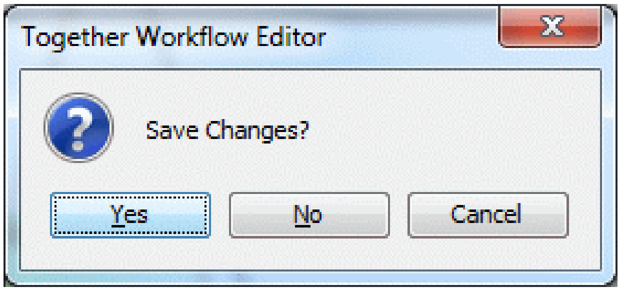
```
#LDAPReferralHandling = follow

#LDAPCountLimit = 0

#LDAPTimeLimit = 0

#LDAPPageSize = 1000

#LDAPHost=localhost

#LDAPPort=389

#LDAPBaseDN=

#LDAPObjectClassFilter=group

#LDAPObjectClassFilterChoices=group,organizationalUnit,organizationalRole,user,person,organizationalPerson,inetOrgPerson,All

#LDAPSearchScope=SCOPE_SUB

#LDAPSecurityLevel=UserAndPassword

#LDAPSecurityUserDN=username@company.com

#LDAPSecurityPassword=somepwd

#LDAPGroupUniqueAttributeName=sAMAccountName

#LDAPGroupNameAttributeName=displayName

#LDAPGroupDescriptionAttributeName=description

#LDAPUserUniqueAttributeName=sAMAccountName

#LDAPUserNameAttributeName=displayName

#LDAPUserDescriptionAttributeName=description

#LDAPDistinguishedNameAttributeName=distinguishedName

#Toolbar.ActionOrder.defaultToolbar = ConfigureLDAP SearchLDAP - ImportSelected ImportAll
```

Chapter 5. Dialogs and Property Panels

Standard Dialogs

Standard dialog are dialogs for manipulation with input/output files.

Save or discard changes dialog

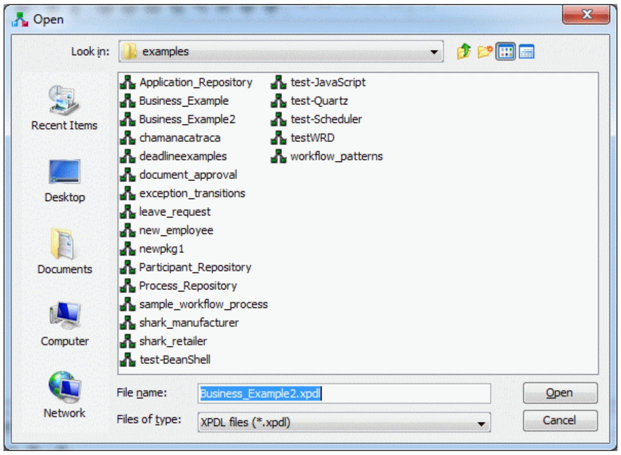


If changes made to the package are about to be discarded, the warning message appears asking if the changes should be saved.

If the answer is "Yes", the current package will be saved. "No" will discard any changes since last save, and "Cancel" will drop out selected function.

File open dialog

This dialog offers file system browsing and choosing file to open. It is displayed when you want to open a new XPDL file for editing in TWE, when adding new external package reference, ...

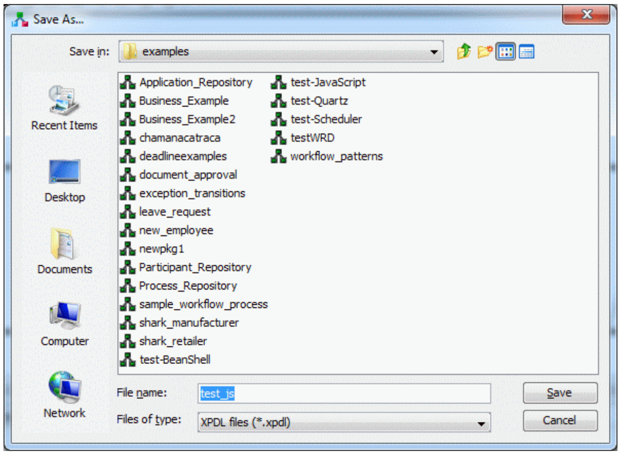


Look in: examples

File name: Business_Example2.xpdl

Files of type: XPDL files (*.xpdl)

Save As dialog



Save in: examples

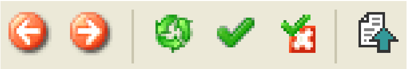
File name: test_js

Files of type: XPDL files (*.xpdl)

This dialog offers file system browsing and choosing the location where you want to save an XPDL file or process or activity set picture, as well as defining the file name and extension.

Dialogs for editing XPDL element properties

The most of the dialogs for editing XPDL elements have a toolbar with a buttons for navigation between property panels, as well as for applying/reverting the changes in the currently displayed property panel.



The toolbar buttons are:

←	Shows previous panel.
→	Shows next panel.
⟲	Discards changes made after last apply action (reverts to previous state).
✓	Applies the changes made in the panel.
☒	Applies the changes made in the panel and closes the dialog.
⇧	Shows parent element panel.

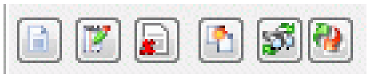
Property panels

Each XPDL element is represented by appropriate property panel for editing its properties.

There are some standard panels like the ones for entering element information through the text box, by choosing an item from a combo box, for managing a group of elements through a table or a list, the group panels containing that contain other panels, tab panels, etc.

Each panel can be displayed separately, or inside a more complex panel such as a group panel.

Property panels that display any kind of a table or a list representing a group of the same kind of XPDL



elements have toolbar with buttons to handle these elements. Elements may be processes, activities, activity sets, participants, transactions, applications, namespaces, workflow variables, type declarations, formal parameters, etc.

The toolbar is displayed just above the table/list of elements:

🗋	Creates new element.
✎	Edits selected element.
✖	Deletes selected element.
⧉	Duplicates selected element.
🔍	Finds all other elements that are referencing selected element. The list of referencing elements is displayed within Search component.
⇄	This action is available only for the FormalParameters and DataFields tables. It converts selected Formal parameter into a Data field object (workflow variable), or vice-versa.

When the element from the property table/list is selected, appropriate event is send to all other TWE components in order to refresh their view, and to show their view of the selected element. For example, if we have displayed a property table panel for handling activities, clicking on table rows, different activities will be displayed in all other TWE components - Graph, Overview, Navigator and Package tree components will mark this activity, and XPDL View component will show the part of XML structure representing selected activity. On the other hand, External package relations, WfXML, Search and Problems component won't update the display.

The toolbar buttons are either enabled or disabled, depending if the action they are representing can or can't be performed on the current selection.

Chapter 6. XPDL Elements

This chapter describes how TWE works, corresponding to WfMC (Workflow Management Coalition) specifications (www.wfmc.org[1]). WfMC provides an interface for workflow process definition. The interface defines a common meta-model for describing the process definition and XML schema for interchange of process definitions - XPDL (XML Process Definition Language). The focus is on XPDL schema elements, and TWE's property panels for editing them.

TWE is a tool for *Process Definition* modelling. The final output of this process modelling is a XPDL output file, which can be interpreted at runtime by the workflow engine(s). TWE accomplished three main goals:

- Graphical representation of process definition
- Export of process definitions to XPDL
- Import of any valid XPDL and its graphical representation

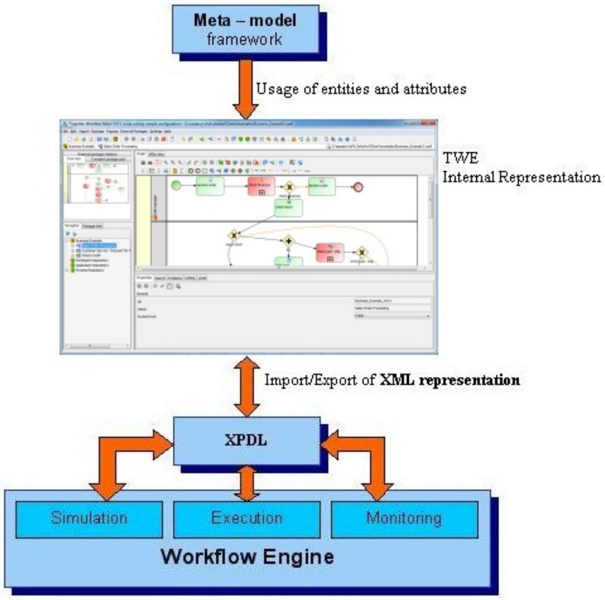
TWE handles one (main) package at a time, although it displays in a read-only mode all the external packages of the main package (and all the external packages of external packages and so on). Main TWE component window is labeled Graph. It is the graphic representation of workflow process or activity set, depending on what is currently selected.

Participants are represented with swim lines which encapsulate other smaller blocks (don't be confused; participants can be horizontally or vertically oriented). Those blocks represent activities.

Lines which connect activities are transitions. Graph specific elements are Start and End bubbles. These elements are not the part of WfMC specification, although they appear in the resulting XPDL file as extended attribute elements.

The rest of XPDL elements can be accessed and modified through their panel properties.

The workflow process definition interface defines a common interchange format, which supports the transfer of workflow process definitions between different products. A workflow process definition, generated by TWE, is capable of interpretation in different workflow run-time products. The principles of *Process Definition Interchange* are based on Meta-Model framework. This meta-data model identifies commonly used entities within a process definition, their relationships and attributes. A variety of attributes describe the characteristics of this limited set of entities. Using this Meta-Model, TWE can transfer models using an XPDL as a common exchange format. Beside this interchange, TWE is also used for internal representation of process definitions. The whole concept is shown on the picture:



There is a mandatory minimum set of objects, which must be supported within XPDL. This "minimum meta-data model" identifies those commonly used entities within a process definition and describes their usage semantics. Extensibility is provided by the facility to encompass additional object attributes ("extended attributes") which can be included as extensions to the basic Meta-Model to meet the specific needs of an individual product or workflow system.

---

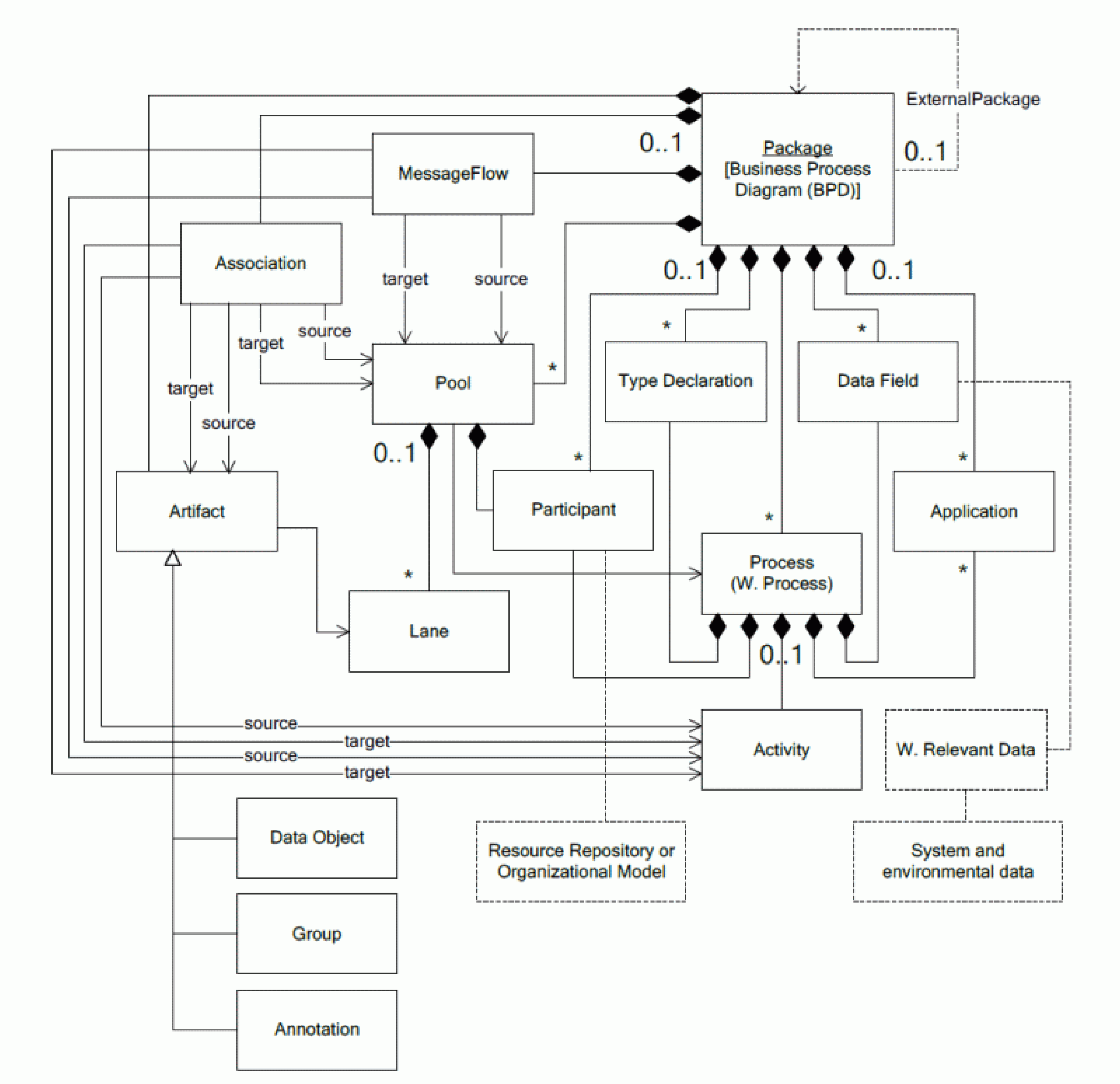
[1] http://www.wfmc.org

Corresponding to the previous picture with concept of Process Definition Interchange, this chapter gives overview of Meta-Model entities (*Meta-Model framework*) and explains representation of these objects in TWE (*Internal representation*) with XPDL attributes (*XML representation*) that are defined in it.

Meta-Model

The meta-model identifies the basic set of entities and attributes for the exchange of process definitions.

Figure 6.1. Package Meta-Model



For each of the above entities, there is an associated set of attributes (some mandatory and others optional) which describe the characteristics of the entity. If there is a need of using an additional characteristics, "extended attributes" for various entities may be defined to allow extension of the scope of the Meta-Model in a controlled manner.

All those entities except "Message Flow" and "Group" type artifacts are maintained by TWE. The current version of TWE is written as an XPDL 1 extension towards XPDL 2, and still does not support the whole XPDL 2 Meta-Model but only the parts necessary to support simple BPMN conformance, and still to allow workflow engines to execute XPDL 1 "migrated" process definitions.

"Resource Repository " is external to the workflow process definitions. In some complex processes, participant declaration may refer to a resource repository, which may be on Organisational Model (OM). WfMC Meta-Model specification defines a simple in-built (Minimal) Organisational Model or permits access to an externally defined OM. Note that XPDL specification does not define or require a resource repository.

TWE works only with a *Minimal Organisational Model* so there is no external Organisational Model provided. In this model, there aren't any relationships between participants. TWE may refer (import) any external XPDL structure - *External Package*. That External XPDL file may contain whole Organisational Model, so that External Package can act like *Resource Repository*.

As it is shown in the previous picture, minimal process model includes various entities whose scope may be wider than a single process definition. In particular the definitions of participants, applications and workflow relevant data may be referenced from a number of process definitions. The Meta-Model assumes the use of a common process definition *Resource Repository*. Repository holds the various entity types comprising the process definition. Within the repository itself and to support the efficient transfer of process definition data to/from the repository, the concept of a PACKAGE is introduced, which acts as a *container* for the grouping of common data entities from a number of different process definitions, to avoid redefinition within each individual process definition. Each process definition contained within the package will automatically inherit any common attribute from the process model container, unless they are separately re-specified locally within the process definition.

Package

It is possible to define several processes within one package, which may share the same tools (applications) and participants.

The Package acts as a container for grouping together a number of individual process definitions and associated entity data, which is applicable to all the contained process definitions.

You can chose to create one package per workflow process, which should contain all the necessary workflow processes as well as all the associated tools (workflow applications) and workflow participants. Another approach is to define just parts of one process definition or common parts of several processes within one package (e.g. a workflow participant list or a workflow application list), and then to reference it from other packages. Within a package, the scope of the definitions of some entities is global and these entities can be referenced from all workflow process definitions contained within the package. Those entities are:

- Workflow Process Definition,
- Workflow Participant Specification,
- Workflow Application Declaration and
- Workflow Relevant Data

TWE provides a way to manage above listed entities (within one package). On Figure 4.3, “Main tool bar shortcut groups”, you can see where is the toolbar part for handling the properties of the selected Package.

To create a new Package in TWE, you simply select File->New, or click appropriate button in the toolbar 🗋 (the first button on the left). If you are currently editing some other XPDL file and there are some unsaved changes there, prior to creating a new Package, TWE will ask you if you want to save the changes.

It is important to say that actions represented by buttons shown on the toolbar picture (and also corresponding menu items) are applied to the currently selected package, which can be the main one or one of the externally referenced packages.

If you select the main package, you can modify its properties, but this is not possible for the external packages. Those external packages are read-only, and you can only read their properties.

Package attributes

Table 6.1. General attributes

Name	**M/O**	**Description**
Id	M (mandatory)	Used to identify the package.
Name	O (optional)	Name of the package.

When you create a new Package in TWE, these attributes are getting a default value newpkg. It's up to user to change it to some meaningful values.

Table 6.2. Package Header

Name	**M/O**	**Description**
XPDL Version	M	Version of XPDL.
Vendor	M	Defines the origin of this package definition and contains vendor's name, vendor's product name and product's release number.
Created	M	Creation date of workflow package.
Description	O	Short textual description of the workflow package.
Documentation	O	Operating System specific path- and filename of help file/description file.
Priority Unit	O	A text string with user defined semantics.
Cost Unit	O	Units used in Simulation Data (usually expressed in terms of a currency).

The Package Header keeps all information central to a package. By default, when you create a new Package, XPDL version will be 2.1 (the version currently partially supported by TWE). If TWE reads a document that have XPDL version set to the value different then 2.1, it will report it as an warning. When new package is created, *Vendor* attribute is set to *Together* and *Created* attribute to the current date and time of creation in ISO-8601 format.

Table 6.3. Redefinable Header

Name	**M/O**	**Description**
Author	O	Name of the author of this package.
Version	O	Version of this package.
Code page	O	The code page used for the text parts.

Name	M/O	Description
Country key	O	Country code based on ISO 3166. It could be either the three digits country code number, or the two alpha characters country codes.
Responsible(s)	O	Workflow participants, that is responsible for this workflow package; the supervisors during run time (usually an Organisational Unit or a Human). It is assumed that the supervisors are responsible during run time.
Publication Status	O	Status of the Package. Possible values are: • UNDER_REVISION • RELEASED • UNDER_TEST

The redefinable header covers those header attributes that may be defined at the package definition header and may be redefined in the header of any process definition.

If publication status is *set to Released*, by default TWE does not allow saving of such XPDL document if it contains any kind of errors or warnings (there is a configuration property *AllowInvalidPackageSaving* in togwecontroller.properties file, which you can change in order to always allow saving of document).

Table 6.4. Conformance Class

Name	M/O	Description
Conformance Class	O	Describes the Conformance Class to which the definitions in this package are restricted to. Possible values are: • FULL-BLOCKED the network structure is restricted to proper nesting of SPLIT/JOIN and LOOP. • LOOP-BLOCKED the network structure is restricted to proper nesting of LOOP. • NON-BLOCKED there is no restriction on the network structure. This is the default.

The conformance class declaration allows description of the conformance class to which the definitions in this package definition are restricted to. The specified class applies to all the contained process definitions. TWE supports all defined conformance classes. This means that when creating graphs (networks of activities and transitions) for a WorkflowProcess or ActivitySet, TWE will report as an error if graph conformance is not satisfied.

Table 6.5. Script element

Name	M/O	Description
Type	M	Identifies the scripting language used in expressions. It is recommended that the Type will be selected from the following strings: text/javascript, text/vbscript, text/tcl, text/ecmascript, text/xml. This
Version	O	This is the version of the scripting language.
Grammar	O	This is a reference to a document that specifies the grammar of the language. It could be, for example, an XML schema, a DTD, or a BNF.

The Script element identifies the scripting language used in XPDL expressions. In the runtime, workflow engine should consider the type specified, and interpret all expressions using appropriate scripting language interpreter. So, after specifying the scripting language type, make sure you are writing all the expressions (for transition conditions, actual parameters, deadline conditions, etc.) according to the language syntax.

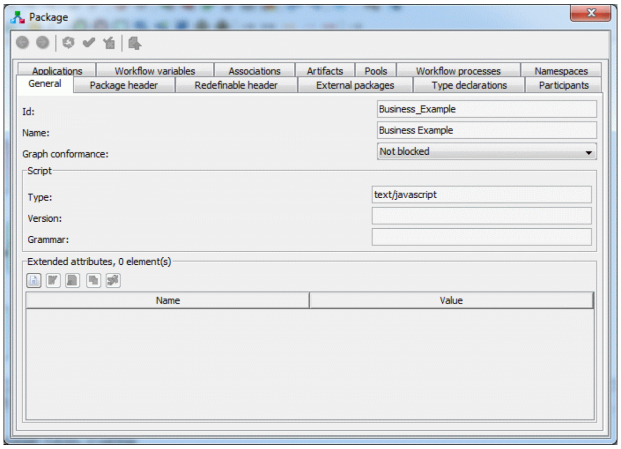
Package property panel

In TWE, the icon from package toolbar (or package menu) is used for defining above mentioned properties, as well as all other package properties and sub-elements, through appropriate property panels. Also, you can get property panel for any Package element by selecting it in Package tree component, and choosing Properties action from the edit menu or toolbar.

Package property panel contains a lot of different data about the main XPDL element - Package. Actually, you are able to access almost any XPDL element panel by navigating through the Package properties panel. All information are organized in several tabs: general, package header, redefinable header, external packages, type declarations, participants, applications, workflow variables, pools, associations, artifacts, workflow processes and namespaces

All the tabs that will be mentioned can be also displayed as a separate property panels.

General tab - displays general package data



Id: Business_Example
Name: Business Example
Graph conformance: Not blocked
Script
Type: text/javascript
Version:
Grammar:
Extended attributes, 0 element(s)

Tab has tree parts. First part contains package's id, name and graph conformance. As already mentioned, TWE supports and validates XPDL depending on specified conformance level.

Second part contains information about script language: type, version and grammar.

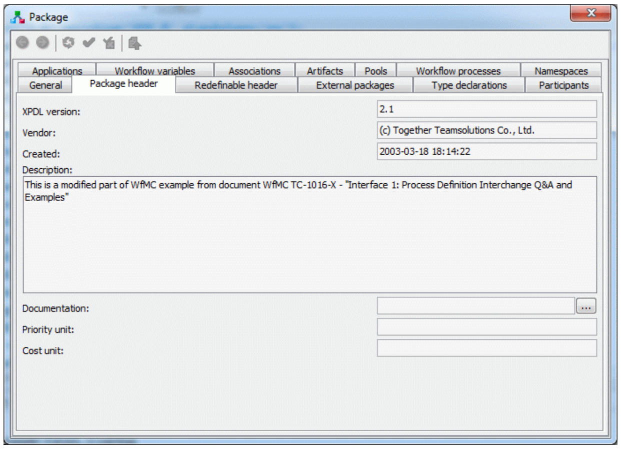
Third part displays list of package's external attributes and offers toolbar with operations to handle extended attributes

Package header tab

Default value for field *XPDL version* is 2.1, and for field *vendor* is (c) Together Teamsolutions Co., Ltd.

Field *Created* contains creation time and date, and field *Description* contains short package's description.

Documentation field binds external file with package. ... button opens a choose file dialog for finding appropriate one.



XPDL version: 2.1

Vendor: (c) Together Teamsolutions Co., Ltd.

Created: 2003-03-18 18:14:22

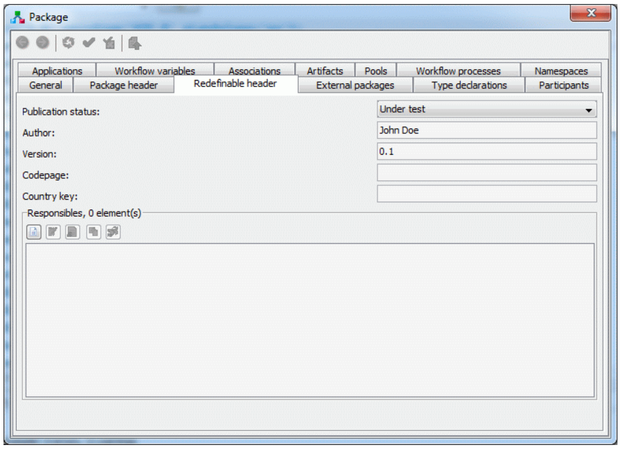
Description: This is a modified part of WfMC example from document WfMC TC-1016-X - "Interface 1: Process Definition Interchange Q&A and Examples"

Documentation:

Priority unit:

Cost unit:

Redefinable header tab



Publication status: Under test

Author: John Doe

Version: 0.1

Codepage:

Country key:

Responsibles, 0 element(s)

Tab has two parts. First part contains the following fields: *Publication status* (can have one of the following values: under revision, under test, or released), *Author* (package's author), *Version* (version number for the package), *Codepage* and *Country key*.

As already mentioned, TWE lets you save your work despite the errors if you didn't set *Publication status* to *Released*. In the case you've set it to released, it won't be permitted to save your work until you correct all the errors. This behaviour can be configured in togwecontroller.properties file, by setting the value of property *AllowInvalidPackageSaving* to true.

Second part, called *Responsibles*, contains list of all responsibles for the package and operations for managing the list. Any participants known to this package can become the responsible.

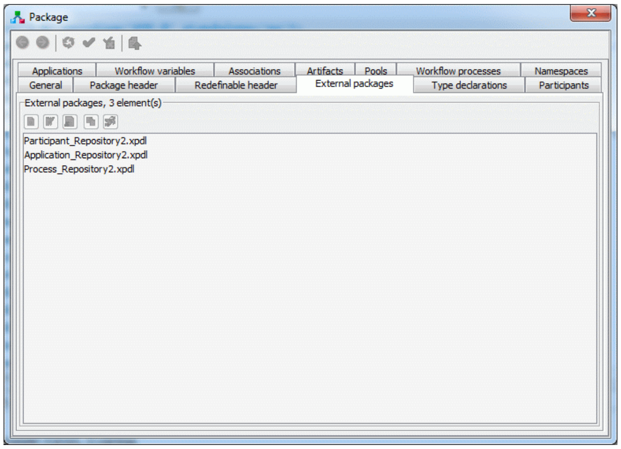
When you press the button for defining new responsible, this action invokes a window with a combo box with a list of all possible participants you can chose.

Responsible person must be a participant that is already defined. When adding "Responsibles" for the whole package, participants defined inside that package or inside externally referenced packages can be added, and when adding "Responsibles" for a process, you can also add participants defined for this process.

Beside the combo box with a list of the responsibles, there is a shortcut button. If you press this button, the property panel for the participant selected in combo box will be shown.

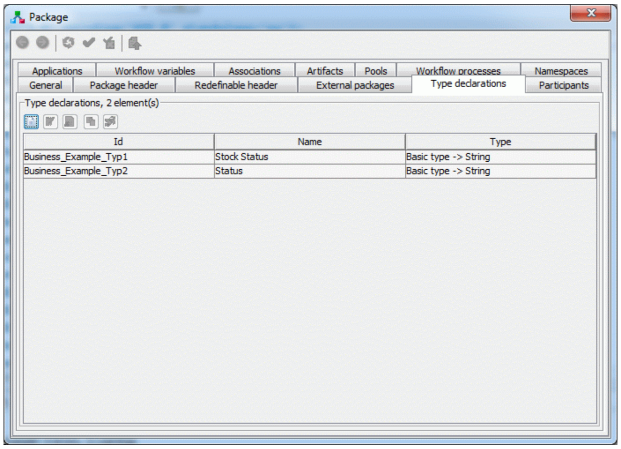
External packages tab

It contains a list of all external packages for the package and toolbar buttons for managing the listed external package elements.



Participant_Repository2.xpdl
Application_Repository2.xpdl
Process_Repository2.xpdl

Type declarations tab

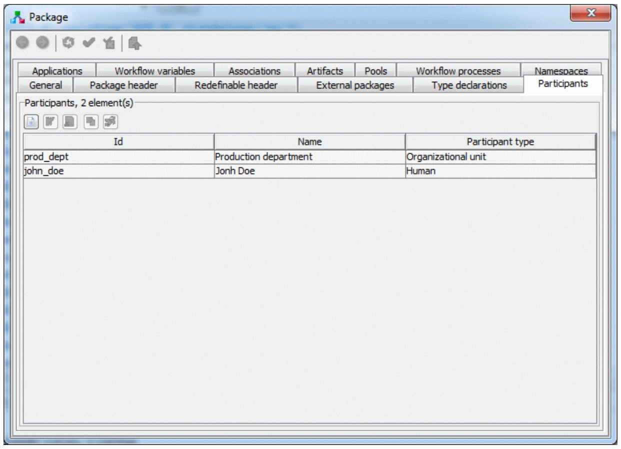


Id	Name	Type
Business_Example_Typ1	Stock Status	Basic type -> String
Business_Example_Typ2	Status	Basic type -> String

It contains information about all type declarations in a form of a table and operations for their managing. Each table row (type declaration) is described with *id*, *name* (optional value) and *type*. It contains toolbar buttons for managing the listed type declaration elements.

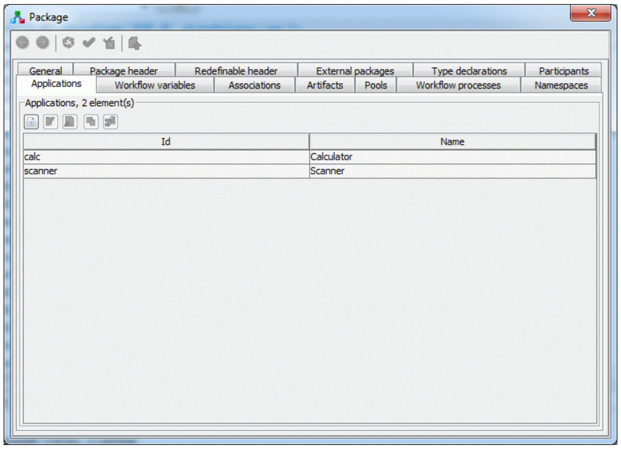
Participants tab

It contains information about all package's participants. Each table row (participant) is described with *id*, *name* (optional value) and *participant type*. It contains toolbar buttons for managing the listed participant elements.



Id	Name	Participant type
prod_dept	Production department	Organizational unit
john_doe	Jonh Doe	Human

Applications tab



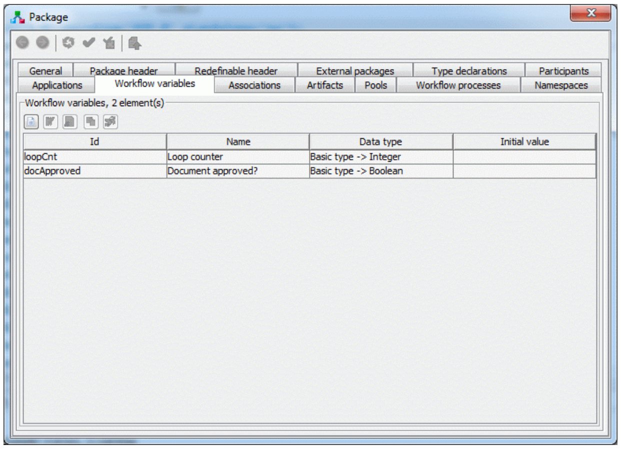
Applications, 2 element(s)

Id	Name
calc	Calculator
scanner	Scanner

It contains information about all package's applications. Each table row (application) is described with *id* and *name* (optional value). It contains toolbar buttons for managing the listed application elements.

Workflow variables tab

It contains information about all package's workflow variables. Each table row (workflow variable) is described with *id*, *name* (optional value) and *data type*. It contains toolbar buttons for managing the listed workflow variable elements.

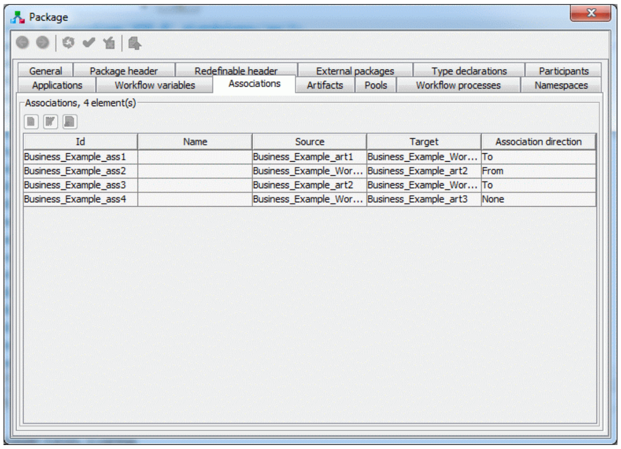


Workflow variables, 2 element(s)

Id	Name	Data type	Initial value
loopCnt	Loop counter	Basic type -> Integer	
docApproved	Document approved?	Basic type -> Boolean	

Associations tab

It contains information about all associations between Artifacts and Activities defined in the Package. Each table row is described with *id*, *name* (optional value) , source and the target of an association and *association direction*. It contains toolbar buttons for managing the listed elements.

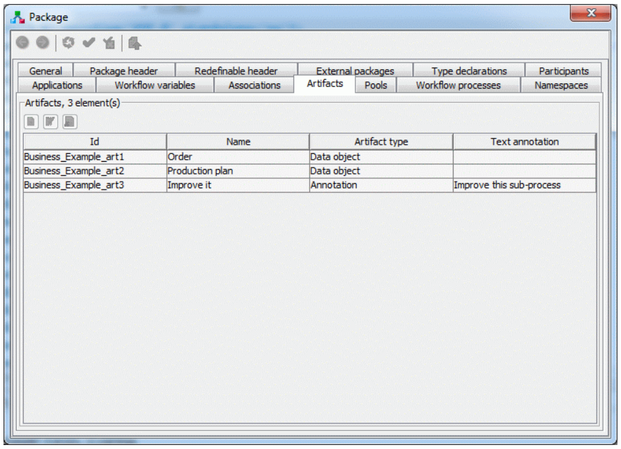


Associations, 4 element(s)

Id	Name	Source	Target	Association direction
Business_Example_ass1		Business_Example_art1	Business_Example_Wor...	To
Business_Example_ass2		Business_Example_Wor...	Business_Example_art2	From
Business_Example_ass3		Business_Example_art2	Business_Example_Wor...	To
Business_Example_ass4		Business_Example_Wor...	Business_Example_art3	None

Artifacts tab

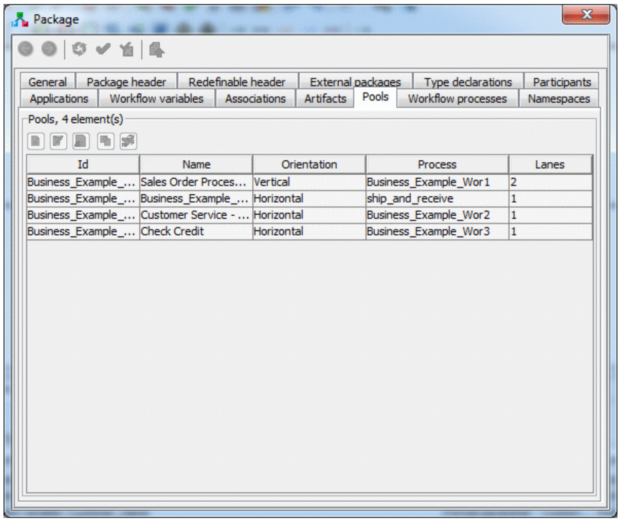
It contains information about all artifacts defined in the package. Each table row is described with *id*, *name* (optional value), artifact type and *text annotation* (optional value). It contains toolbar buttons for managing the listed elements.



Artifacts, 3 element(s)

Id	Name	Artifact type	Text annotation
Business_Example_art1	Order	Data object	
Business_Example_art2	Production plan	Data object	
Business_Example_art3	Improve it	Annotation	Improve this sub-process

Pools tab

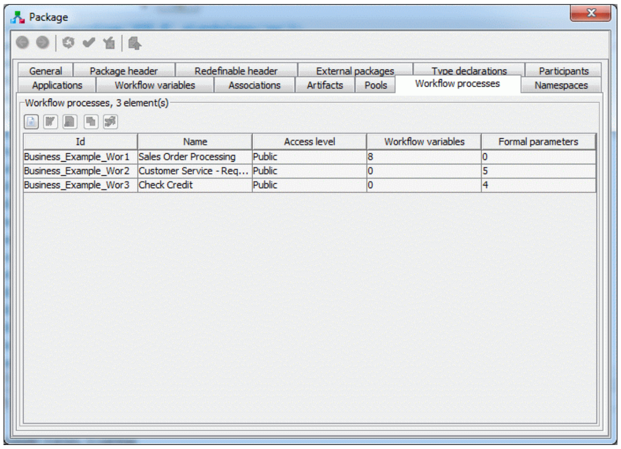


Pools, 4 element(s)

Id	Name	Orientation	Process	Lanes
Business_Example_...	Sales Order Proces...	Vertical	Business_Example_Wor1	2
Business_Example_...	Business_Example_...	Horizontal	ship_and_receive	1
Business_Example_...	Customer Service - ...	Horizontal	Business_Example_Wor2	1
Business_Example_...	Check Credit	Horizontal	Business_Example_Wor3	1

It contains information about all the package's pools for workflow processes. Each table row (pool) is described with *id*, *name* (optional value), orientation, process (a reference to the process or activity set) and *Lanes*. TWE does not allow creation or removal of the pools. Whenever new Workflow process is created or removed, corresponding pool gets created/removed. There is a toolbar where only the button that allows to edit the pool properties is enabled.

Workflow processes tab



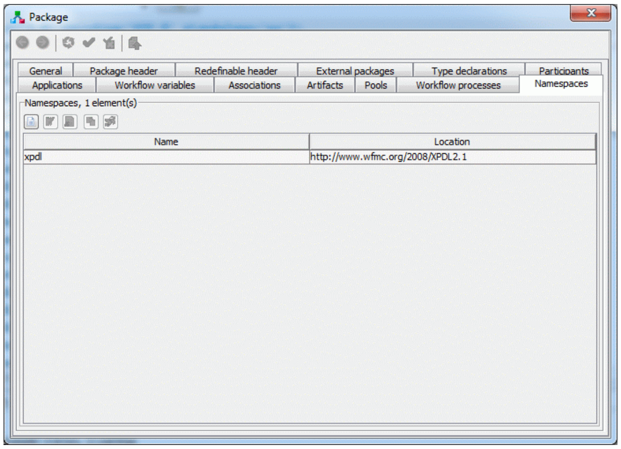
Workflow processes, 3 element(s)

Id	Name	Access level	Workflow variables	Formal parameters
Business_Example_Wor1	Sales Order Processing	Public	8	0
Business_Example_Wor2	Customer Service - Req...	Public	0	5
Business_Example_Wor3	Check Credit	Public	0	4

It contains information about all package's workflow processes. Each table row (workflow process) is described with *id*, *name* (optional value) and *Access level* (optional value, can have one of the values: *private* or *public*). It contains toolbar buttons for managing the listed workflow process elements.

Namespaces tab

This property panel shows all namespaces defined within the XPDL document. User can add additional namespaces. Usually, additional namespaces are needed to insure document validity when user defines "complex" extended attributes. It contains information about all package's namespaces in a form of a table and operations for their managing. Each table row (namespace) is described with *name* and *location*. It contains toolbar buttons for managing the listed namespace elements.



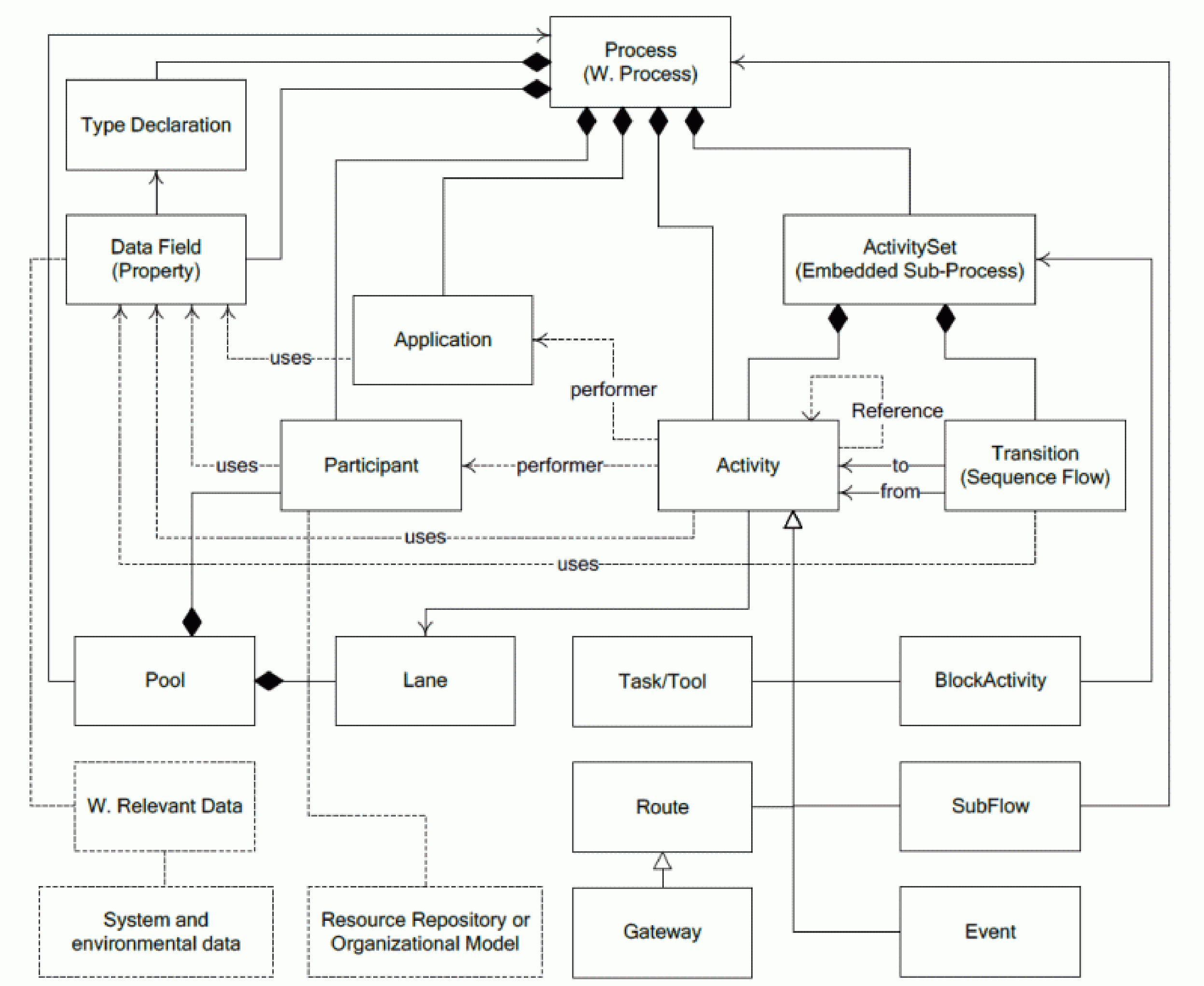
Namespaces, 1 element(s)

Name	Location
xpdl	http://www.wfmc.org/2008/XPDL2.1

Workflow Process

The Process Definition entity provides contextual information that applies to other entities within the process. This describes the process itself. It is a container for the process itself and provides information associated with administration (creation date, author, etc.) or to be used during process execution (initiation parameters to be used, execution priority, time limits to be checked, person to be notified, simulation information, etc.). Definition entity thus provides header information for the process definition and is therefore related to all other entities in that process.

Figure 6.2. Workflow Process Meta-Model



The Workflow Process Definition defines the elements that make a workflow. It contains definitions or declarations, respectively, for Activity and, optionally, for Transition, Application, and Process Relevant Data entities.

A Workflow Process may run as an implementation of an activity of type subflow; in this case parameters may be defined as attributes of the process.

TWE provides a way to manage process definition entities. On Figure 4.3, “Main tool bar shortcut groups”, you can see where is the toolbar part for handling the propertiesOn Figure 4.3, “Main tool bar shortcut groups” you can see the organization of the selected WorkflowProcess.

If you create (override) some process with the same Id as the one from the external package, only the one from the current package can be used as the implementation process of sublfow activity.

Typically, you will create a new process by pressing the button from the main toolbox or selecting appropriate item from Package menu.

WorkflowProcess attributes

WorkflowProcess attributes can be divided in few logical parts:

Table 6.6. General Attributes

Name	M/O	Description
Id	M	Used to identify the workflow process (Read Only).
Name	O	Name of the model, used to identify the workflow process.
AccessLevel	O	The Access level of a process may be either PUBLIC or PRIVATE. If PUBLIC, the process may be invoked by an external system or application. A process with private access may only be invoked from a SubFlow Activity.

Table 6.7. Process Header

Name	M/O	Description
Duration Unit	M	Describes the default unit to be applied to an integer duration value that has no unit tag. Possible units are: Y - year M - month D - day H - hour m - minute s - second
Created	O	Creation date of workflow process definition.
Description	O	Short textual description of the process.
Priority	O	The priority of the process type. Default: Inherited from Model Definition.
Limit	O	Expected duration for time management purposes (e.g. starting an escalation procedure etc.) in units of DurationUnit. It is counted from the starting date/time of the Process. The consequences of reaching the limit value are not defined in this document (i.e. vendor specific). It is assumed that in this case at least the Responsible of the current process is notified of this situation.
Valid From	O	The date that the workflow process definition is active from. Empty string means system date. Default: Inherited from Model Definition.

Name	M/O	Description
Valid To	O	The date at which the process definition becomes valid. Empty string means unlimited validity. Default: Inherited from Model Definition.
Waiting Time	O	Describes the amount of time, which is needed to prepare the performance of the task (time estimation) (waiting time is provided by the analysis environment and may be updated by the runtime environment) in units of DurationUnit.
Working Time	O	Describes the amount of time the performer of the activity needs to perform the task (time estimation) (working time is needed for analysis purposes and is provided by the evaluation of runtime parameters) in units of DurationUnit.
Duration	O	Expected duration time to perform a task in units of DurationUnit.

In Process Header Tab there is Time Estimation group where we define Waiting Time, Working Time and Duration. This is used for simulation purposes.

Table 6.8. Redefinable Header

Name	M/O	Description
Publication Status	O	Status of the Workflow Process Definition. Default: Inherited from Model Definition. UNDER_REVISION RELEASED UNDER_TEST
Author	O	Name of the author of this workflow process definition. (the one who puts it into the package)
Version	O	Version of this workflow process definition.
Codepage	O	The codepage used for the text parts. Default: Inherited from Model Definition.
Country key	O	Country code based on ISO 3166. It could be either the three digits country code number, or the two alpha characters country codes. Default: Inherited from Model Definition.

Name	M/O	Description
Responsible(s)	O	Workflow participant, who is responsible for this workflow process (usually an Organisational Unit or a Human). It is assumed that the supervisor is responsible during execution of the process. Default: Inherited from Model Definition.

The responsibles for Process are added in the same way as it is at Package level, which is explained earlier in text.

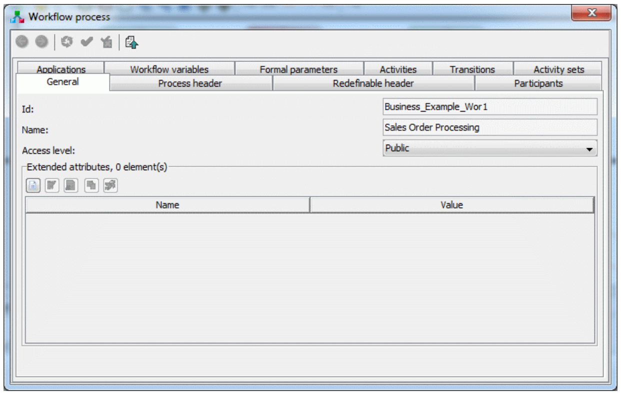
Workflow Process property panel

In TWE, you can get the property panel to edit all workflow process attributes by clicking on a toolbar icon , or selecting the appropriate menu item.

It contains a lot of different data about workflow process. All information are organized in several tabs: general, process header, redefinable header, participants, applications, workflow variables, formal parameters, activities, transitions and activity sets.

All the tabs that will be mentioned can be also displayed as a separate property panels, either by selecting appropriate element in a Package tree, and asking for its properties, or by selecting appropriate toolbox button or menu item.

General tab - displays general process data



Id: Business_Example_Wor1

Name: Sales Order Processing

Access level: Public

Tab has two parts. First part contains process *id*, *name* (this name will appear in window's title bar) and *access level* (can be private or public).

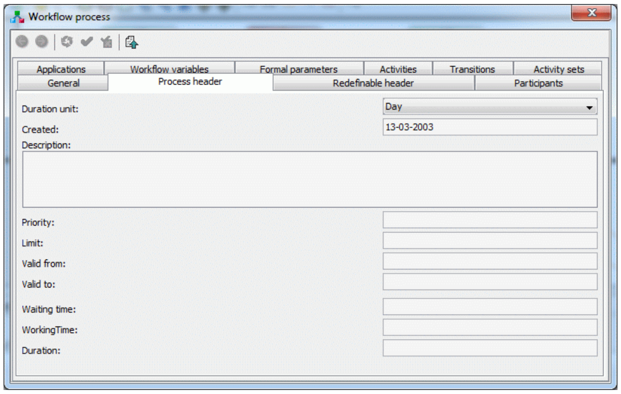
Second part shows all process extended attributes and also offers operations with them.

Process header tab

This dialog defines elements and attributes of XPDL's *ProcessHeader* element.

Field *Duration unit* can have one of the following values: second, minute, hour, day, month, year.

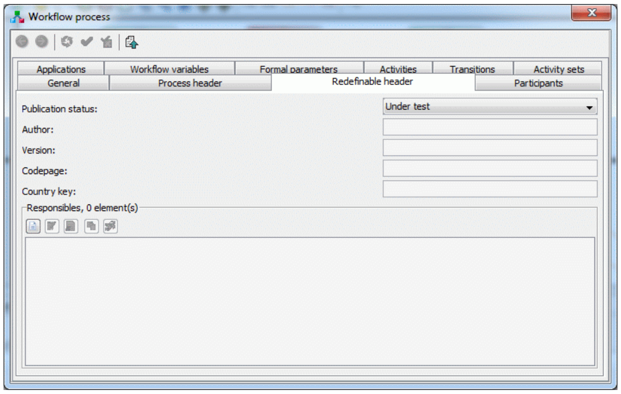
Field *Created* displays the process creation date and field *Description* is place for short description of this workflow process.



Duration unit: Day

Created: 13-03-2003

Redefinable header tab



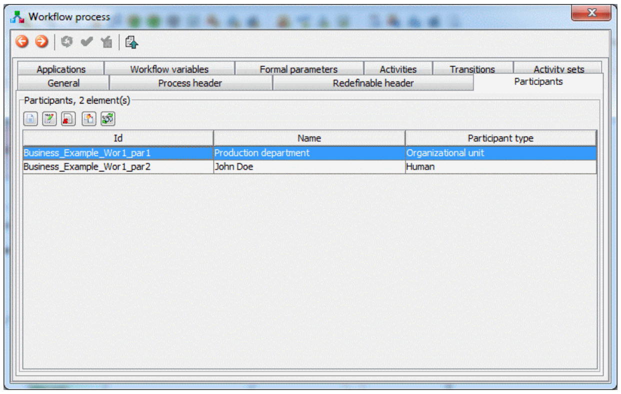
Publication status: Under test
Author:
Version:
Codepage:
Country key:
Responsibles, 0 element(s)

Tab has two parts. First part contains the following fields: *Publication status* (can have one of the following values: under revision, under test, or released), *Author* (author of the process), *Version* (version number for the process), *Codepage* and *Country key*.

Second part, called *Responsibles*, contains list of all responsibles for the process and operations for managing the list.

Participants tab

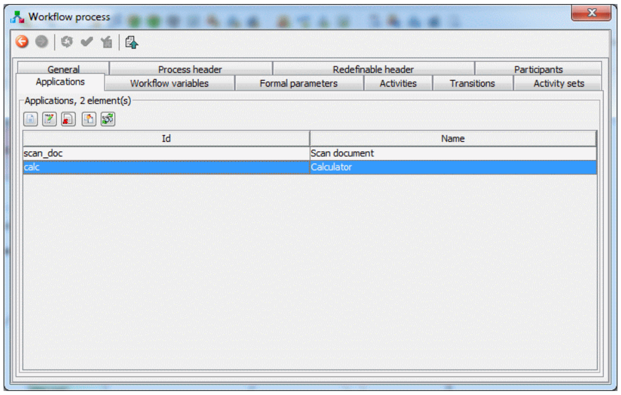
It contains information about all participants of the workflow process. Each table row (participant) is described with *id*, *name* (optional value) and *participant type*. It contains toolbar buttons for managing the listed participant elements.



Participants, 2 element(s)

Id	Name	Participant type
Business_Example_Wor1_par1	Production department	Organizational unit
Business_Example_Wor1_par2	John Doe	Human

Applications tab



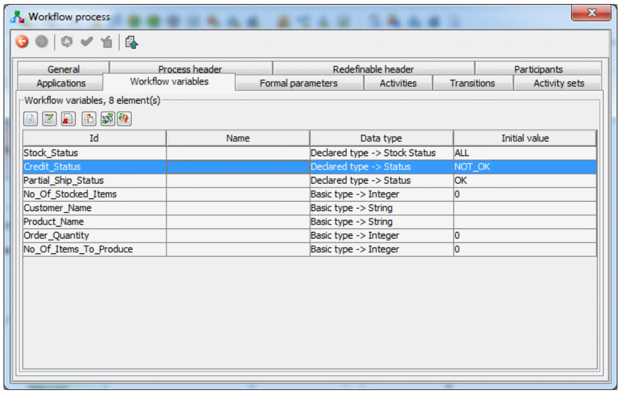
Applications, 2 element(s)

Id	Name
scan_doc	Scan document
calc	Calculator

It contains information about all applications of the workflow process. Each table row (application) is described with *id* and *name* (optional value). It contains toolbar buttons for managing the listed application elements.

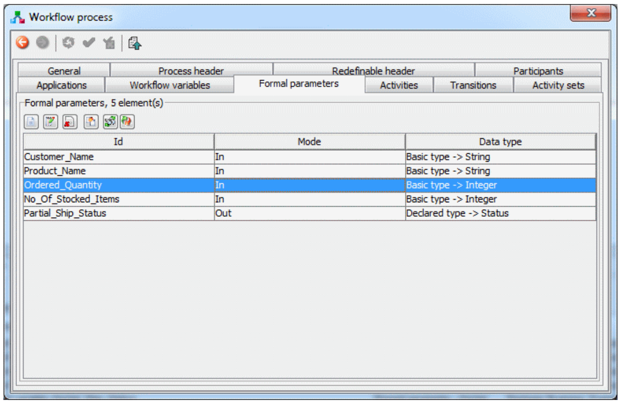
Workflow variables tab

It contains information about all workflow process's workflow variables. Each table row (workflow variable) is described with *id*, *name* (optional value) and *data type*. It contains toolbar buttons for managing the listed workflow variable elements.



Id	Name	Data type	Initial value
Stock_Status		Declared type -> Stock Status	ALL
Credit_Status		Declared type -> Status	NOT_OK
Partial_Ship_Status		Declared type -> Status	OK
No_Of_Stocked_Items		Basic type -> Integer	0
Customer_Name		Basic type -> String	
Product_Name		Basic type -> String	
Order_Quantity		Basic type -> Integer	0
No_Of_Items_To_Produce		Basic type -> Integer	0

Formal parameters tab

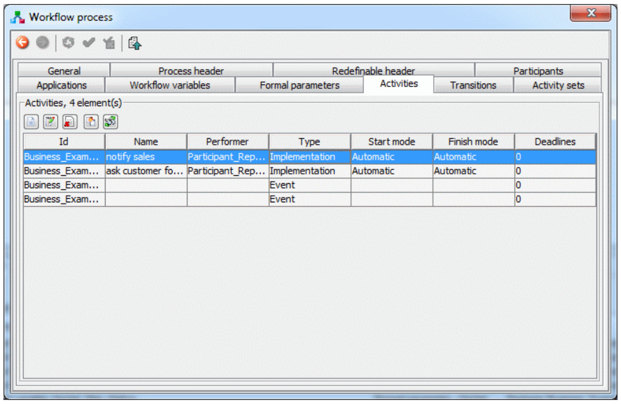


Id	Mode	Data type
Customer_Name	In	Basic type -> String
Product_Name	In	Basic type -> String
Ordered_Quantity	In	Basic type -> Integer
No_Of_Stocked_Items	In	Basic type -> Integer
Partial_Ship_Status	Out	Declared type -> Status

It contains information about all workflow process's formal parameters. Each table row (formal parameter) is described with *id*, *mode* and *data type*. It contains toolbar buttons for managing the listed formal parameter elements.

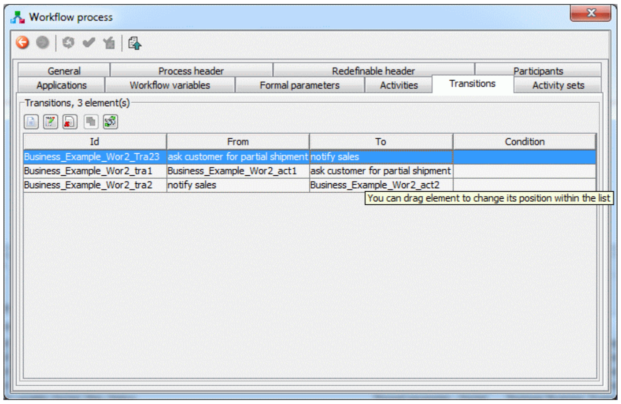
Activities tab

It contains information about all activities of the workflow process. Each table row (activity) is described with *id*, *name* (optional value) and *performer* (optional value). It contains toolbar buttons for managing the listed activity elements.



Id	Name	Performer	Type	Start mode	Finish mode	Deadlines
Business_Exam...	notify sales	Participant_Rep...	Implementation	Automatic	Automatic	0
Business_Exam...	ask customer fo...	Participant_Rep...	Implementation	Automatic	Automatic	0
Business_Exam...			Event			0
Business_Exam...			Event			0

Transitions tab



Transitions, 3 element(s)

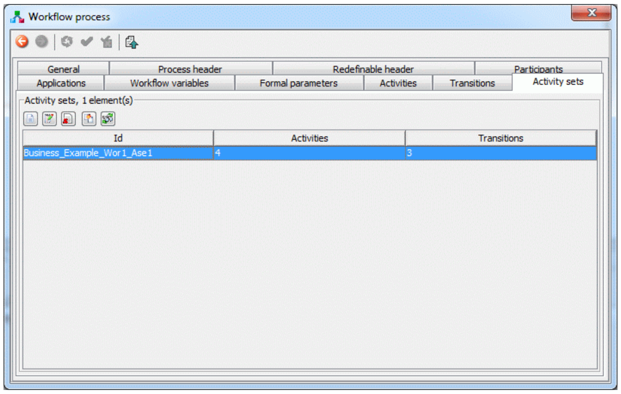
Id	From	To	Condition
Business_Example_Wor2_Tra23	ask customer for partial shipment	notify sales	
Business_Example_Wor2_tra1	Business_Example_Wor2_act1	ask customer for partial shipment	
Business_Example_Wor2_tra2	notify sales	Business_Example_Wor2_act2	

You can drag element to change its position within the list

It contains information about all workflow process's transitions. Each table row (transition) is described with *id*, from (source), to (target) and *condition* (optional value). It contains toolbar buttons for managing the listed transition elements.

Activity sets tab

It contains information about all activity sets of the workflow process. Each table row (activity set) is described with *id*, *activities* and *transitions*. It contains toolbar buttons for managing the listed activity set elements.

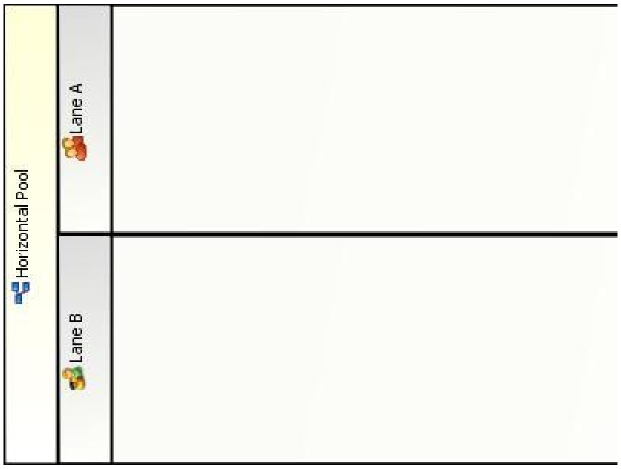


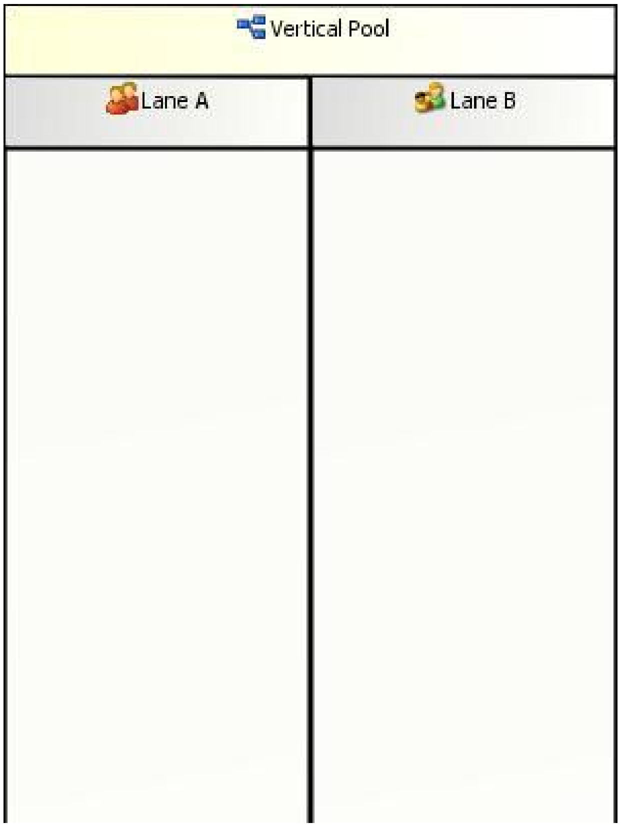
Activity sets, 1 element(s)

Id	Activities	Transitions
Business_Example_Wor1_Ase1	4	3

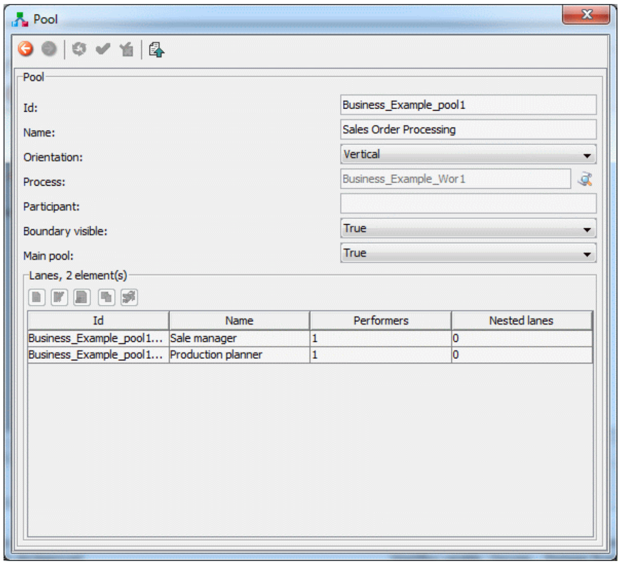
Swimlanes (Pools and Lanes)

Swimlanes are used to facilitate the graphical layout of a collection of processes and the activities they contain. They may designate participant information at the process level and performer information at the activity level. The Swimlane structure is depicted by a collection of non-overlapping rectangles called Pools. Each Pool may be further subdivided into a number of Lanes.





Pools



Id: Business_Example_pool1
Name: Sales Order Processing
Orientation: Vertical
Process: Business_Example_Wor1
Participant:
Boundary visible: True
Main pool: True

Id	Name	Performers	Nested lanes
Business_Example_pool1...	Sale manager	1	0
Business_Example_pool1...	Production planner	1	0

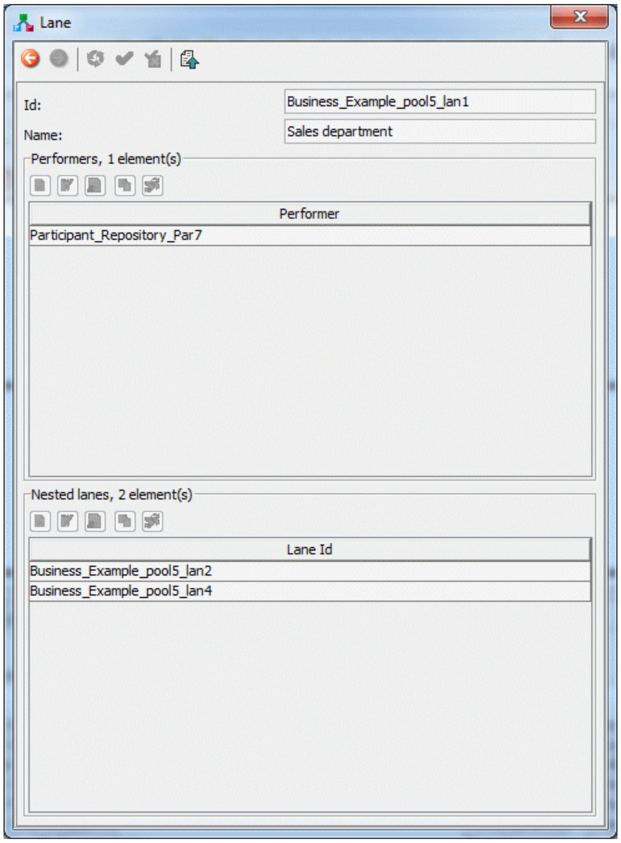
A Pool acts as the container for flow objects (activities and artifacts) and Sequence Flow (transitions and associations) between them.

Together Workflow Editor does not allow users to create or remove a pool. Pool is automatically being created or deleted whenever workflow process gets created/deleted.

The property panel for the pool shows the pool attributes which are editable, except the text field that displays the Id of referenced Workflow Process/Activity Set, and has only the link button to display the properties of the referenced Workflow Process/Activity Set.

Lanes

Lanes are used to subdivide a Pool. All the activities within a Lane may inherit one or more properties from the Lane. A typical use of this is to give the Lanes "role names" and have the Activities inherit these role names as "Participant assignment/Performer expressions".



Id: Business_Example_pool5_lan1

Name: Sales department

Performers, 1 element(s)

Performer
Participant_Repository_Par7

Nested lanes, 2 element(s)

Lane Id
Business_Example_pool5_lan2
Business_Example_pool5_lan4

Together Workflow Editor does not allow users to create lanes using properties dialogs. The only way to create or delete a lane is by using TWE's Graph component. Lane is always being created as a representation of the Participant. Lanes are implicitly created whever user wants to insert a lane representation for the existing or a new participant into the graph. When activity is put into the lane (using graph component), the activity's performer becomes the participant which is a representation for that particular lane.

As described in the section called “Graph” , TWE also alows nesting of lanes.

The property dialog of the lane shows lane properties Id, Name, Performers and Nested Lanes. The Id and the name are editable fields. The performer of the lane is also editable. You can change the performer of the lane to link it with an another participant, or you can write your own expression for the performer. which are editable (including a performer of the lane), but you can not add or the remove the performer of the lane. The Nested Lanes can't be added or removed as well (it is possible only through the Graph component).

External References

External package concept allows you to re-use common Participant, Application and WorkflowProcess definitions contained within other Package definitions. You just need to define these common entities ones, and store it within a XPDL file, and then re-use it afterwards in all other XPDLs you create

TWE provides possibility of referencing external Packages. After referencing it, automatically, every participant, application, type declaration or workflow process defined there in will be accessible from the main package.

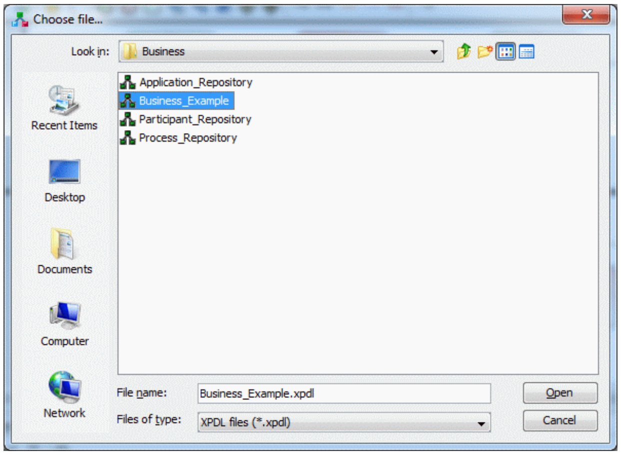
If external package we referenced also contains the references to other external packages, all of the package references will be imported in TWE in read-only mode.

TWE's *External package relations component* will show the relations of the main package external packages, their external packages, and so on.

Adding External Packages

This dialog appears when you press the package toolbar button, or Package menu item for adding external packages. It enables you to select the XPDL file from your disk, and to reference it as an external package for the package you are currently editing. As already mentioned, after adding external package, you are able to use participants, applications and workflow processes defined in that package.

NOTE: You can't add external package if the main XPDL package you are working on is not yet saved to the disc.



Look in: Business

Application_Repository
Business_Example
Participant_Repository
Process_Repository

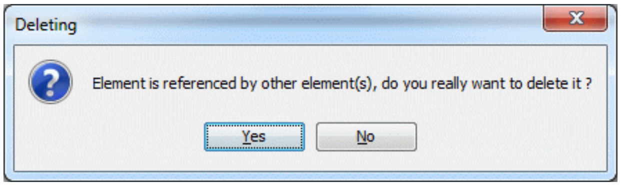
File name: Business_Example.xpdl

Files of type: XPDL files (*.xpdl)

Removing External Packages

Action *Remove external packages* from the toolbox, or External packages menu removes external package from the current package.

If this external package is not referenced by any other external package of the main package, it will be really removed from TWE's memory. In the case some of the elements from this external packages are in use (Applications, Participants or WorkflowProcesses), you'll be asked for the confirmation.



Element is referenced by other element(s), do you really want to delete it ?

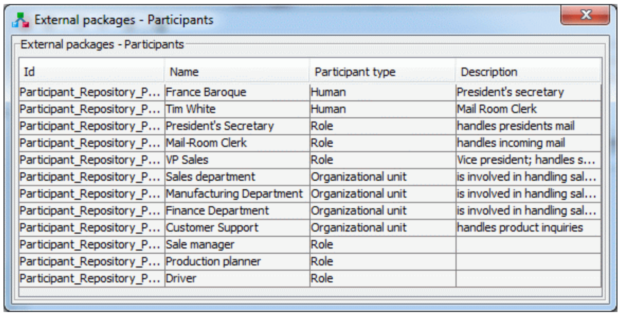
The external package mechanism allows packages and its elements (applications, processes and participants) to be multiple referenced.

Before the removement of the selected external package, if it contains any element that is referenced, and the external package reference that will be deleted is the last one, the warning message about removement of the external package is displayed.

External Participants

To get the following dialog, click button (or appropriate menu item from External Packages menu).

This dialog shows all participants of package's external packages. Every external participant is presented with its *Id*, *Name* (optional data), *Participant Type* and *Description* (optional data).

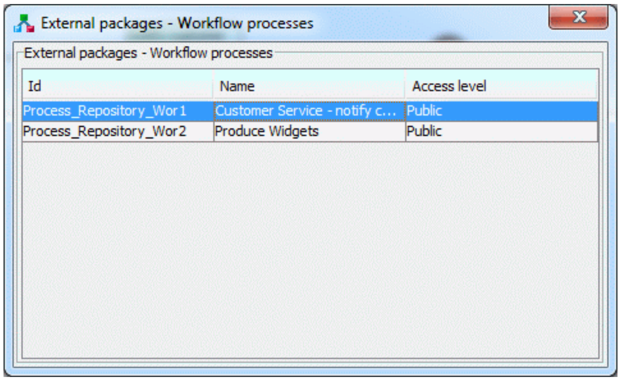

External packages - Participants
External packages - Participants

Id	Name	Participant type	Description
Participant_Repository_P...	France Baroque	Human	President's secretary
Participant_Repository_P...	Tim White	Human	Mail Room Clerk
Participant_Repository_P...	President's Secretary	Role	handles presidents mail
Participant_Repository_P...	Mail-Room Clerk	Role	handles incoming mail
Participant_Repository_P...	VP Sales	Role	Vice president; handles s...
Participant_Repository_P...	Sales department	Organizational unit	is involved in handling sal...
Participant_Repository_P...	Manufacturing Department	Organizational unit	is involved in handling sal...
Participant_Repository_P...	Finance Department	Organizational unit	is involved in handling sal...
Participant_Repository_P...	Customer Support	Organizational unit	handles product inquiries
Participant_Repository_P...	Sale manager	Role	
Participant_Repository_P...	Production planner	Role	
Participant_Repository_P...	Driver	Role	

External Processes

To get the following dialog, click button (or appropriate menu item from External Packages menu).

This dialog shows all workflow processes of package's external packages. Every external process is presented with its *Id*, *Name* (optional data) and Access level (optional data).

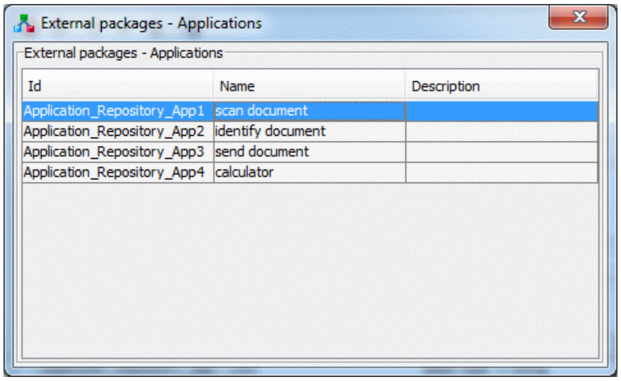

External packages - Workflow processes
External packages - Workflow processes

Id	Name	Access level
Process_Repository_Wor1	Customer Service - notify c...	Public
Process_Repository_Wor2	Produce Widgets	Public

External Applications

To get the following dialog, click button (or appropriate menu item from External Packages menu).

This dialog shows all applications of package's external packages. Every external application is presented with its *Id*, *Name* (optional data) and *Description* (optional data).

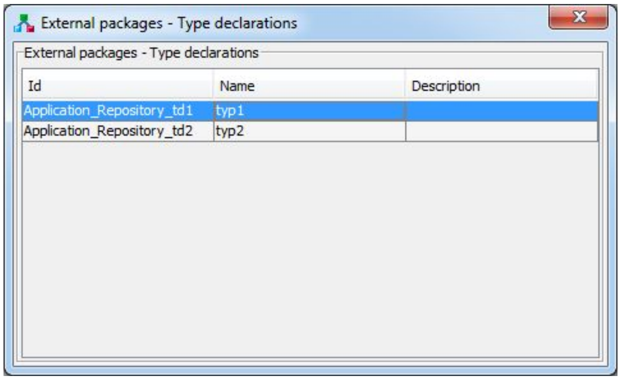

External packages - Applications
External packages - Applications

Id	Name	Description
Application_Repository_App1	scan document	
Application_Repository_App2	identify document	
Application_Repository_App3	send document	
Application_Repository_App4	calculator	

External Type Declarations

To get the following dialog, click button (or appropriate menu item from External Packages menu).

This dialog shows all type declarations of package's external packages. Every external type declaration is presented with its *Id*, *Name* (optional data) and *Description* (optional data).


External packages - Type declarations
External packages - Type declarations

Id	Name	Description
Application_Repository_td1	typ1	
Application_Repository_td2	typ2	

Type Declaration

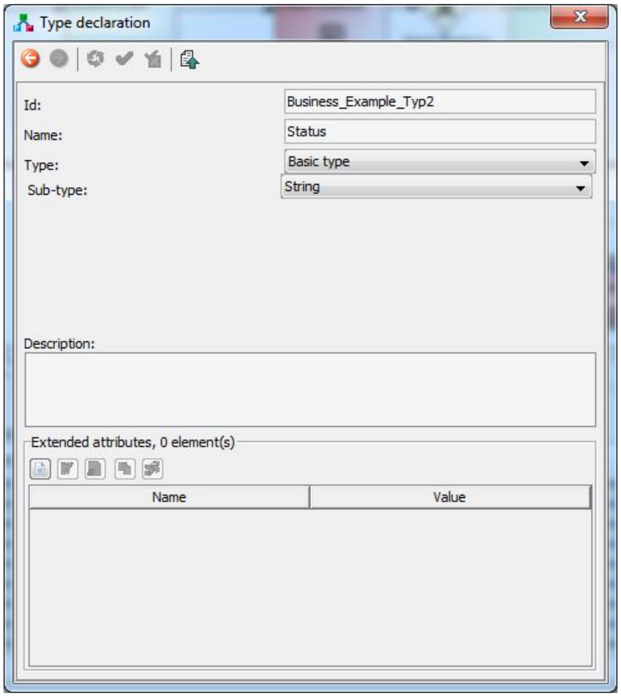
WfMC assumes a number of standard data types (string, reference, integer, float, date/time, etc.); such data types are relevant to workflow relevant data, system or environmental data or participant data. Expressions may be formed using such data types to support conditional evaluations.

Sometimes set of data types that XPDL provides won't be enough, or you want to represent some data type with a special name to easily use it when defining Formal/Actual parameters. This XPDL feature allows you to declare new data type.

Attributes for TypeDeclaration are:

Table 6.9. Type Declaration

Name	M/O	Description
Id	M	Used to identify the declared data type.
Name	O	Text used to identify the Declared Data Type.
Description	O	Short textual description of the Declared Data Type.
Data Types	O	Standard data types.



Id: Business_Example_Typ2
Name: Status
Type: Basic type
Sub-type: String
Description:

This property panel shows an example of a type declaration.

This panel can be get e.g. by selecting an existing TypeDeclaration in a Package tree, and asking for its properties. To create a new TypeDeclaration, you can press button ▦ which will bring you the property panel with all defined Type declarations for the package. There in, you can press a button for creating new Type declaration.

TWE maintains all of WfMC data types that are provided: BasicType, DeclaredType, ShemaType, ExternalReference, RecordType, UnionType, EnumerationType and ArrayType.

The following table describes the basic data type:

Table 6.10. Basic Data Type

Name	M/O	Description
Type	M	STRING

Name	M/O	Description
		FLOAT INTEGER REFERENCE DATETIME BOOLEAN PERFORMER

External Reference data type has the following attributes:

Table 6.11. External Reference

Name	M/O	Description
Xref	O	It specifies the identity of the entity within the external document.
Location	M	It specifies the URI of the document that defines the type.
Namespace	O	It allows specification of the scope in which the entity is defined.

Using External Reference data type you may define e.g. some Java class as a new data type (for e.g. `location = "com.abc.purchases.PO"`).

Workflow Participant

WfMC Meta-Model specification defines a simple in-built (Minimal) Organisational Model or permits access to an externally defined OM. Participants in TWE are just part of an Organisational Model - Minimal OM. The connection with the Organisational Model is used in Activity Definition (performer of an activity) and in the Process Definition (responsible of a process).

Workflow Participants have a scope and visibility equivalent to extended attributes. All referenced Workflow Participants have to be defined in the scope where they are used, at least in the same package.

The Workflow Participant is defined by a type and related information, which is a set of type specific attributes. This definition contains a basic set of 6 Workflow Participant types: *resource set, resource, organizational unit, role, human,* or *system*. A role and a resource are used in the sense of abstract actors. This definition is an abstraction level between the real performer and the activity, which has to be performed. During run time these abstract definitions are evaluated and assigned to concrete human(s) and/or program(s).

These attributes are used to define a Workflow Participant:

Table 6.12. Workflow Participant

Name	M/O	Description
Id	M	Used to identify the workflow participant definition.
Participant Name	O	Text used to identify a performer.
Participant Type	M	Definition of the type of workflow participant entity. Type of a workflow participant.
Participant Description	O	Short textual description of a workflow participant.

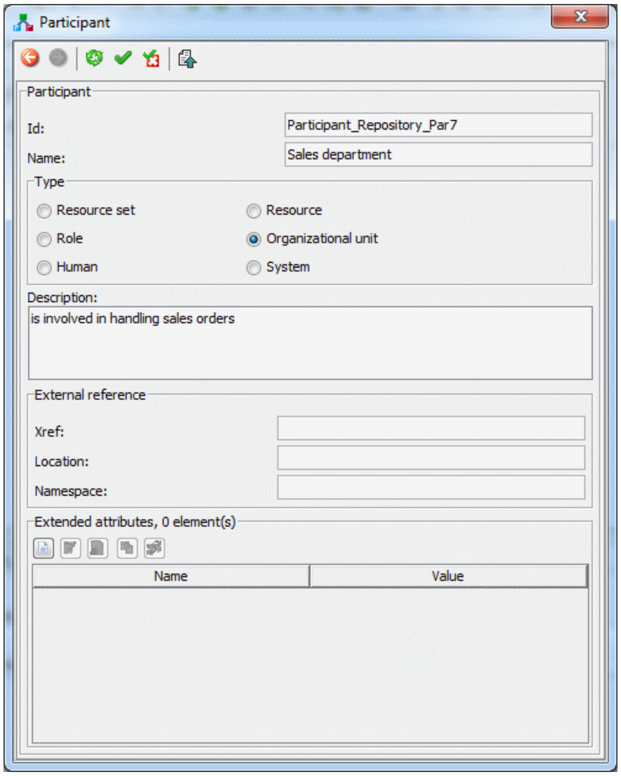
Name	M/O	Description
External Reference	O	A reference to an external specification of a participant.

The Participant entity type attribute characterizes the participant to be an individual, an organisational unit or an abstract resource such as a machine. Here is a list of possible participant types:

Table 6.13. Participant Types

Name	Description
RESOURCE_SET	A set of resources.
RESOURCE	A specific resource agent (as a machine).
ROLE	This type allows performer addressing by a role or skill set. A role in this context is a function a human has within an organization. As a function isn't necessarily unique, a coordinator may be defined (for administrative purposes or in case of exception handling) and a list of humans the role is related to.
ORGANIZATIONAL_UNIT	A department or any other unit within an organizational model.
HUMAN	A human interacting with the system via an application presenting a user interface to the participant.
SYSTEM	An automatic agent.

TWE provides the way to define participants at Package level and at Process Definition level. In order to do that, you can select appropriate toolbar button or menu item for showing all the participants defined at package/process level. This will bring a table property panel, where you'll have a button for creating a new participant.



Participant

Id: Participant_Repository_Par7

Name: Sales department

Type

○ Resource set ○ Resource

○ Role ● Organizational unit

○ Human ○ System

Description: is involved in handling sales orders

External reference

Xref:

Location:

Namespace:

Extended attributes, 0 element(s)

Name	Value

You are able to insert the lanes representing all the defined participants into a TWE graph using the following choice button from graph's toolbar , and it'll be represented as a new swim-lane in a graph. Then, you can put the activities (and artifacts) in there, which actually means that activity's performer is the participant represented by the swim-lane.

Even if you do not define any participant, TWE's graph component has special lanes that can be used to define activity's performer as an expression (can be created by toolbar buttons and).

All the activities which performer is not a reference to an existing Participant will be graphically displayed inside those special graph swim-lanes.

There is another way of defining Package level Participants, and this is by using a Graph's toolbox button.

Pressing this button , and than clicking in the graph will automatically add another participant represented as a new swim-lane in currently selected WorkflowProcess's graph. This participant's type will be Role, and you can change it through the property panel.

Workflow Application

These are some of the attributes for application entity which are defined by WfMC specification:

Table 6.14. General attributes

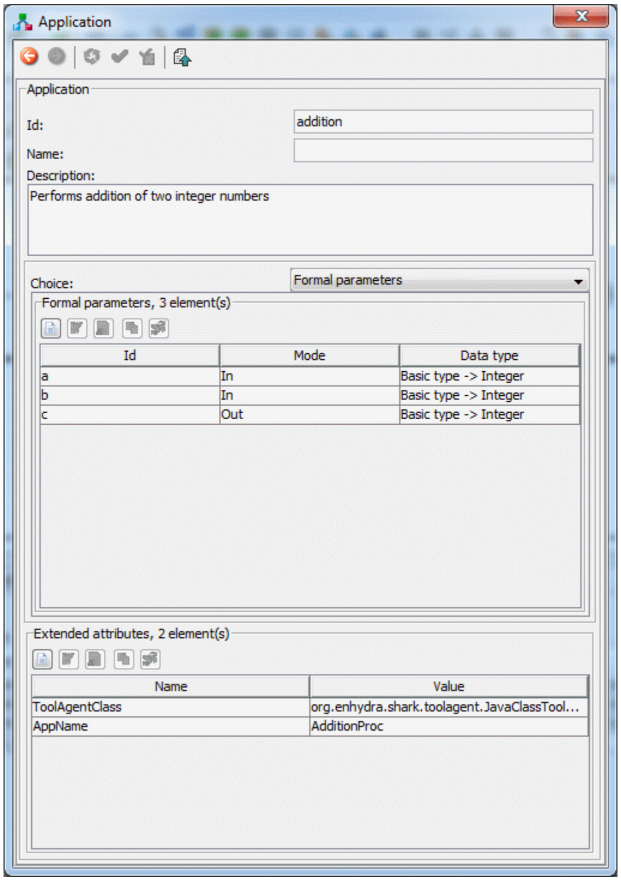
Name	**M/O**	**Description**
Id	M	Used to identify the workflow application definition.
Name	M	Text used to identify an application (may be interpreted as a generic name of the tool).
Description	O	Short textual description of the application.
Extended attributes	O	Optional extensions to meet individual implementation needs.
Invocation Parameters	O	Parameters that are interchanged with the application via the invocation interface.

There is more than one way of getting *Workflow Application Declaration* settings. One way would be choosing an icon at main toolbar. Another way would be selecting Package's *Applications* in a Package tree component, and then right clicking and selecting Properties action (or directly selecting this action from Edit menu).

User can create new instance for application entity, edit some existing application or delete it (modification and deletion of application entity is not allowed if it is the entity from externally referenced package). The picture on the right shows a property panel for editing application attributes.

Workflow applications that are defined for the package are accessible by any activity that is defined at any package's workflow process.

When defining a Tool for an activity, you'll be able to chose amongst all applications defined inside the particular WorkflowProcess definition, Package definition, or inside definition of the Package's external packages. If the application from the process level has the same Id as the one from the package level (it overrides the one from the package level), the one from the package level won't be displayed. The same stands for overriding the application from external package.



Id: addition

Name:

Description:
Performs addition of two integer numbers

Choice: Formal parameters

Formal parameters, 3 element(s)

Id	Mode	Data type
a	In	Basic type -> Integer
b	In	Basic type -> Integer
c	Out	Basic type -> Integer

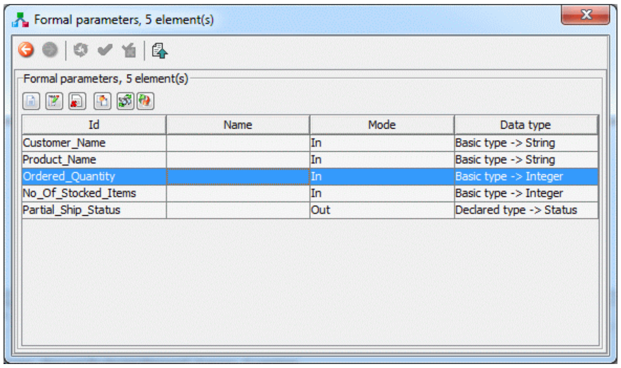
Extended attributes, 2 element(s)

Name	Value
ToolAgentClass	org.enhydra.shark.toolagent.JavaClassTool...
AppName	AdditionProc

As it is shown on the picture, there are two choices for Invocation Parameters:

- Formal Parameters and
- External Reference

Formal Parameters


Formal parameters, 5 element(s)
Formal parameters, 5 element(s)

Id	Name	Mode	Data type
Customer_Name		In	Basic type -> String
Product_Name		In	Basic type -> String
Ordered_Quantity		In	Basic type -> Integer
No_Of_Stocked_Items		In	Basic type -> Integer
Partial_Ship_Status		Out	Declared type -> Status

Formal Parameters are parameters that are interchanged with the application via the invocation interface. They are passed during invocation and returned of control (e.g. of an invoked application or invoked sub-process).

The order of formal parameters can be changed by dragging an item with the mouse.

Using appropriate toolbar buttons, you can create a new FormalParameter, edit, delete or duplicate selected one, as well as to get all the references to the selected FormalParameter.

The attributes that define Formal Parameter are:

Table 6.15. Formal Parameter

Name	M/O	Description
Id	M	Identifier for the parameter
Mode	M	• IN Input Parameters • OUT Output Parameters • INOUT Parameters used as input and output
Name	O	Name of the parameter
Is Array	O	If it is an array type
Data Type	O	Data type of the formal parameter
Initial value	O	Initial value for parameter
Description	O	Textual description of the formal parameter
Length	O	The length of the data.

External Reference

An External Reference can be used instead of formal parameters. External Reference is a reference to an external definition of an entity.

External Reference has the following attributes:

Table 6.16. External Reference

Name	M/O	Description
Xref	O	It specifies the identity of the entity within the external document.

Name	**M/O**	**Description**
Location	M	It specifies the URI of the document that defines the type.
Namespace	O	It allows specification of the scope in which the entity is defined.

With External Reference, Application (and some other entities) may be defined by XML schema (for e.g. `location = http://abc.com/schemas/po.xsd`), by a Java class (for e.g. `location = "com.abc.purchases.PO"`), by WSDL document (for e.g. `location = "http://abc.com/services/poService.wsdl"`)...

Workflow Relevant Data

Workflow Relevant Data (or, in XPDL, Data Field) represent the variables of a Process Definition or Package Definition. They are typically used to maintain decision data (used in conditions) or reference data values (parameters), which are passed between activities or sub-processes. The workflow relevant data list defines all data objects, which can be used within the workflow process. The attribute `TYPE` explicitly specifies all information needed for a workflow management system to define an appropriate data object for storing data, which is to be handled, by an active instance of the workflow process.

Workflow Relevant Data can be defined in a Workflow Process (the Workflow Process Relevant Data) and in a Package (the Package Relevant Data). The scopes differ in that the former may only be accessed by entities defined inside that process, while the latter may be accessed by entities inside any process defined within that model.

Workflow Relevant Data has a scope that is defined by the directly surrounding meta-model entity and is not nested. The visibility of its identifier is also defined by that entity.

Attributes of Workflow Relevant Data are:

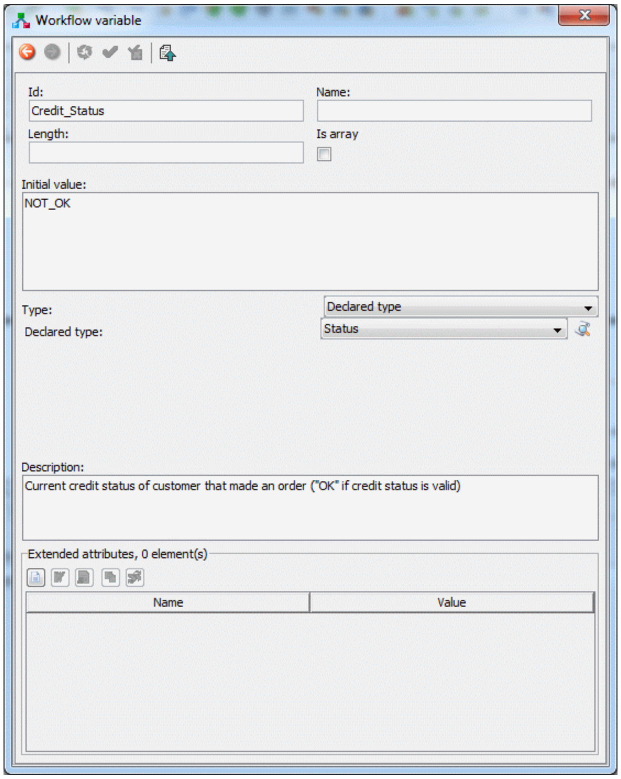
Table 6.17. Relevant Data

Name	**M/O**	**Description**
Id	M	Used to identify the workflow relevant data.
Data Type(s)	M	Data Type.
Name	O	Text used to identify the workflow relevant data.
Is Array	O	Indicates if it is an array.
Initial Value	O	Pre-assignment of data for run time.
Length	O	The length of the data.
Description	O	Short textual description of the defined data.

When parameters are passed to a called subflow outside the current model definition, it is the responsibility of the process designer(s) to ensure that data type compatibility exists across the parameter set.

Workflow relevant data naming must be unique within a process model. If such data is passed between processes as parameters, the convention at this version of specification is that copied semantics will be used. Responsibility rests with process designers / administrators to ensure consistent name / datatype usage within process definitions / models to support sub-flow operations (including any required remote process interoperability).

There is an icon in Package's part of the Toolbar and icon in process's part of the toolbar, and also on appropriate menus, for displaying all workflow relevant data for the selected Package/WorkflowProcess.



Id: Credit_Status
Name:
Length:
Is array ☐
Initial value: NOT_OK
Type: Declared type
Declared type: Status
Description: Current credit status of customer that made an order ("OK" if credit status is valid)
Extended attributes, 0 element(s)

Name	Value

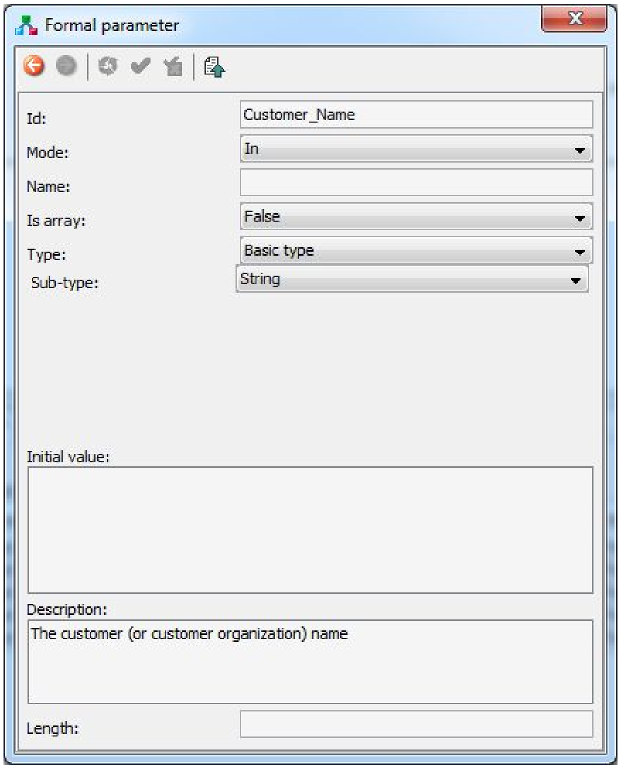
Like other similar table property panels, amongst others, there are also buttons "New" and "Edit" for adding new or editing existing Workflow Relevant Data. The picture on the left shows property panel for editing Workflow Relevant Data.

When defining (or editing) an workflow variable, the following attributes can be set: *id* (mandatory attribute), *name*, *initial value* , length, *IsArray* (array indicator), *Type*, *description* and *extended attributes*.

The attribute *Type* explicitly specifies all information needed for a workflow management system to define an appropriate data object for storing data, which is to be handled, by an active instance of the workflow process.

If you are at process level, and you create (override) some workflow relevant data with the same Id as the one from the package level, only one of them (the one from the process level) can be used as a reference from other entities. E.g. you can choose just the one from the process level to be the actual parameter for some application or subflow.

Formal Parameters



Id: Customer_Name
Mode: In
Name:
Is array: False
Type: Basic type
Sub-type: String
Initial value:
Description: The customer (or customer organization) name
Length:

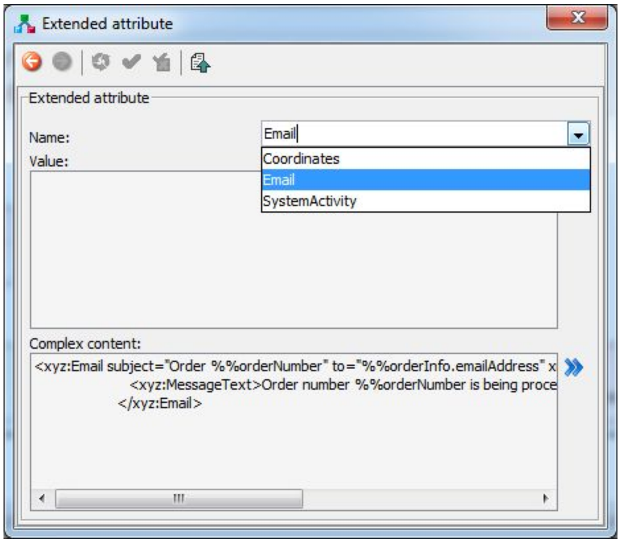
Formal Parameters are parameters that are interchanged with the application via the invocation interface. They are passed during invocation and return of control (e.g. an invoked application or invoked sub-process).

Every formal parameter is presented with its *Id*, *Mode, Data type and Description*.

To add new FormalParameter for the process, you can click on the button (the right-most button in the toolbar). This will bring you a table property panel with a list of all FormalParameters already defined for the selected process. There is a New button there in, which will create a new FormalParameter, and will bring a property panel for edit it.

Extended Attributes

XPDL contains most of the entities, which are likely to be required in the process definition modelling. Sometimes, there is a need for some additional information (user or vendor specific). Extended Attributes are the primary method to support such extensions. These are attributes defined by the user or vendor, where necessary, to express any additional entity characteristics.



Name: Email

Coordinates
Email
SystemActivity

Value:

Complex content:

```
<xyz:Email subject="Order %%orderNumber" to="%%orderInfo.emailAddress" x
              <xyz:MessageText>Order number %%orderNumber is being proce
           </xyz:Email>
```

TWE provides use of the Extended Attributes like follows: you can define "simple" part of Extended Attribute by entering *Name* and *Value* attributes. If you want to define "complex" part of Extended Attribute, which could be consisted of tags that belong to the XPDL or some other namespace, you have to enter it as a free-text in the *Complex content* field.

The picture on the left shows property panel representing extended attribute.

When defining new ExtendedAttribute (or modifying one already defined), user has possibility to choose its name amongst the names of extended attributes already defined for that element type (Activity, Participant, ...). The list of names depends on the names already defined in opened XPDL (and its externally referenced XPDLs). The list of names is refreshed each time new ExtendedAttribute is being added or when one already defined is being re-defined or deleted.

Activity Set

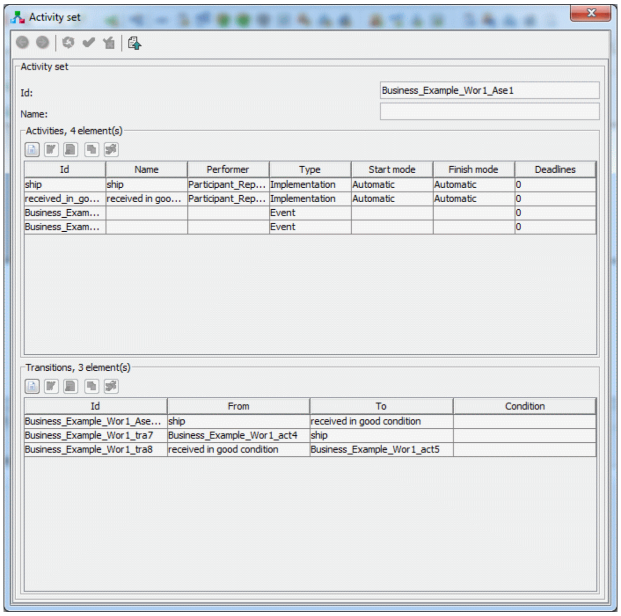
An activity set is a self-contained set of activities and transitions. Transitions in the set should refer only to activities in the same set and there should be no transitions into or out of the set. Activity sets can be executed by block activities.

Typically, you will create a new activity set in the WorkflowProcess using Graph component's toolbar icon for doing that. Previously you must insure that the graph for the WorkflowProcess where you want to insert the activity set is visible.

The picture on the right shows property panel for editing ActivitySet.

Every activity set is presented with its *Id*, *activities* (number of activities in the activity set) and *transitions* (number of transitions in the activity set).

As you can see, there is only one attribute to define, and that is the Id. There are two tables where you can see short description of all activities and transitions already defined for this ActivitySet. As with other table panels, you are able to manage them.



Id: Business_Example_Wor1_Ase1

Name:

Activities, 4 element(s)

Id	Name	Performer	Type	Start mode	Finish mode	Deadlines
ship	ship	Participant_Rep…	Implementation	Automatic	Automatic	0
received_in_go…	received in goo…	Participant_Rep…	Implementation	Automatic	Automatic	0
Business_Exam…			Event			0
Business_Exam…			Event			0

Transitions, 3 element(s)

Id	From	To	Condition
Business_Example_Wor1_Ase…	ship	received in good condition	
Business_Example_Wor1_tra7	Business_Example_Wor1_act4	ship	
Business_Example_Wor1_tra8	received in good condition	Business_Example_Wor1_act5	

Activity

Generally, any process comprises a number of steps, which lead towards the overall goal. Workflow process consists of a number of workflow activities. The workflow activity is a piece of work that will be done by combination of resources and computer applications.

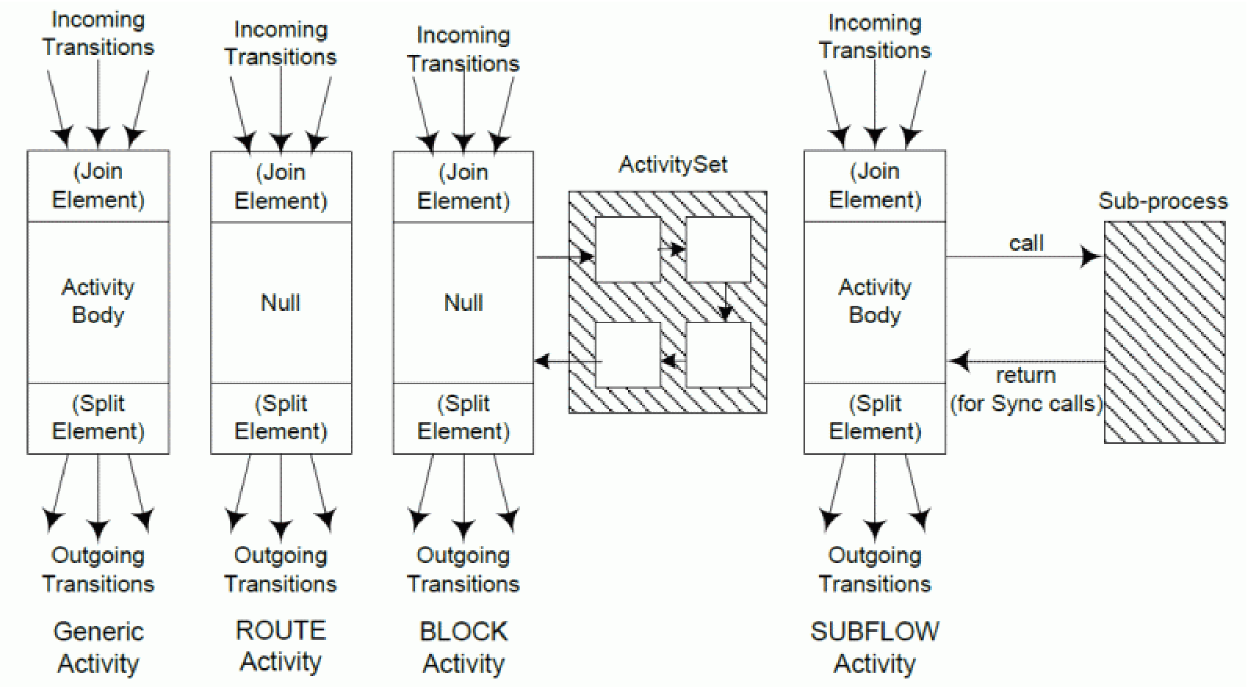
Table 6.18. Activity attributes

Name	M/O	Description
Name	O	Text used to identify process activity.
Performers	O	Link to entity workflow participant. May be an expression. Default: Any Participant.
Start mode	O	Describes how the execution of an Activity is triggered.
Finish mode	O	Describes how the system operates at the end of the Activity.
Deadline	O	Specification of a deadline and action to be taken if it is reached.
Priority	O	A value that describes the initial priority of this activity when the execution starts. If this attribute is not defined but a priority is defined in the Process definition then that is used. By default it is assumed that the priority levels are the natural numbers starting with zero, and that the higher the value the higher the priority (e.g.: 0, 1, ...).
Limit	O	Expected duration for time management purposes (e.g. starting an escalation procedure etc.) in units of DurationUnit. It is counted from the starting date/time of the Process. The consequences of reaching the limit value are not defined in this document (e.g. vendor specific).
Icon	O	Address (path- and filename) for an icon to represent the activity.
Documentation	O	The address (e.g. path - and filename) for a help file or a description file of the activity.
Description	O	Textual description of the activity.

Activities are associated with their performers (which are workflow participants), and application assignments. Optional information about activity may be associated with: starting and stopping manner, usage of specific workflow relevant data, preconditions for starting and postconditions for finishing the activity.

The following diagram illustrates the generic structure of activities:

Figure 6.3. Generic Structure of Activities

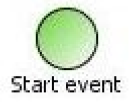
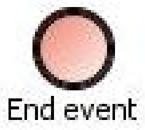
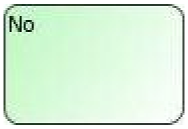
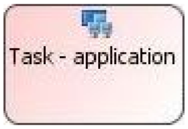


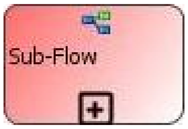
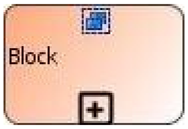
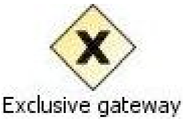
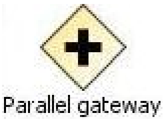
Activities and other activity-like objects are inserted using buttons on graph's "Toolbox" toolbar.

Activities and Graph component

Although it is possible to create a new activity through the property panels, the usual way to do it is through the graph component.

The following table shows the picture of how the graph represents a certain XPDL activity type, and short description of each type of the activity:

	Start event activity - the place where the process begins.
	End event activity - the place where the process ends.
	Manual (No implementation) activities are atomic (Generic Activities). They are the smallest units of work, although even this activity may produce more than one work item for its performer.
	Task-Application activities are atomic (Generic Activity). They are the smallest units of work.

	Subflow is another activity type. It implements a whole new workflow process. Process definition within the subflow is entirely independent from the first one (where subflow activity resides). It has its own set of activities, internal transitions, participants, application definitions and other workflow relevant data. When clicking on the box with the "plus" sign, the graph of the sub-flow gets shown.
	An activity may be a block activity that executes an activity set , or map of activities and transitions. Activities and transitions within an activity set share the name space of the containing process. When clicking on the box with the "plus" sign, the graph of the activity set gets shown.
	Dummy (route) activity does nothing on its own. This type of activities is used for exclusive synchronization and constructing complex and sophisticated "exclusvie" transitional conditions
	Dummy (route) activity does nothing on its own. This type of activities is used for parallel synchronization and parallel (unconditional) branching

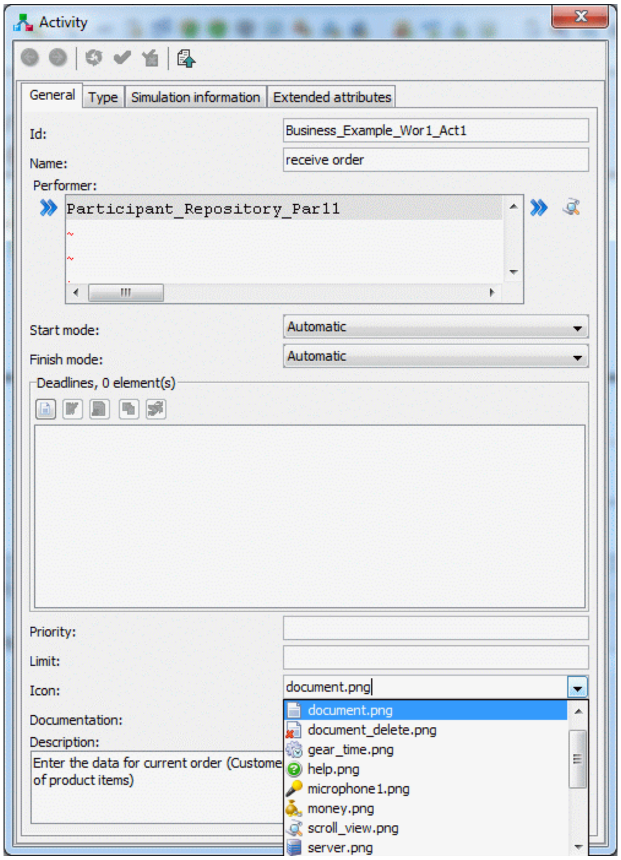
Icons for inserting different types of activities from the graph toolbox are the same as the ones included in the previously explained picture of the activities in the graph. Graph's toolbox icon for inserting Start event activity into the graph is 🟢 and for inserting End event activity into is 🔴. For the manual activity, there is a toolbox icon ▢, for Tool activities ⧉, for subflow activities ⧉, for block activities ⧉ and for route activities ◈ (for exclusive gateway route activity) and ⊕ (for parallel gateway route activity). Once selected, mouse cursor will show what type of object you'll insert. Activity is created using some default values for their properties, which typically are to be changed. Right clicking the object in the graph shows a context menu, and there is item representing action for getting property panel. Different than others, Block and SubFlow activities have additional menu item "descend into..." which is used to display the graph of the referenced ActivitySet/Workflow Process.

Beside this, TWE offers a special feature to select an Icon for particular activity. When you open activity's property panel, for the Icon entry you can select some of the additional Icons we offer (this list can be easily extended by putting more icons in tweactivityicons.jar file). The selected icon will appear instead the default one.

Activity property panel

In this section are explained various property panels concerning different types of activities: activities without implementation (manual), sub-flow, route, tool and block activities. All information are organized in several tabs: general, type, transition restriction, simulation information, and extended attributes.

General tab - displays general activity data



Id: Business_Example_Wor1_Act1

Name: receive order

Performer: Participant_Repository_Par11

Start mode: Automatic

Finish mode: Automatic

Deadlines, 0 element(s)

Priority:

Limit:

Icon: document.png

Documentation:

Description: Enter the data for current order (Custome
of product items)

Tab has tree parts. First part contains activity *id*, *name*, *performer* (can be a reference to a participant or an expression), *start mode* (can be automatic or manual) and *finish mode* (can be automatic or manual).

Second part shows all deadlines and also offers toolbar buttons with options to handle list of deadlines.

Third part contains *priority* (describes initial priority for activity), *limit* (expected duration - for time management purposes), *icon* (reference to the icon representing this activity), *documentation* (reference to an external document) and *description*.

Fields icon and documentation have drop down list to select one of the offered icons. If selected, this icon will replace the original one representing this activity within the graph.

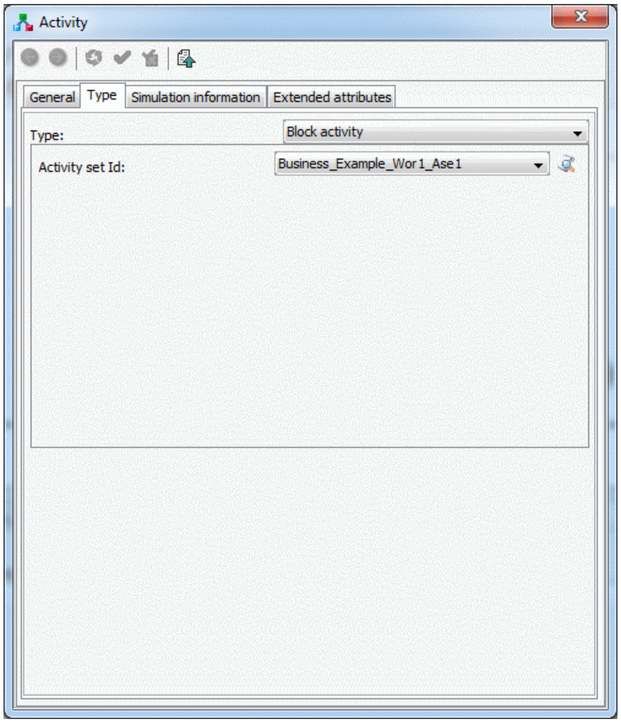
(the list of available icons can be extended by putting additional icons within the tweactivityicons.jar file)

Type tab

This tab contains some specific data concerning the particular type of an activity. For activities without implementation (manual) this tab doesn't contain any specific data.

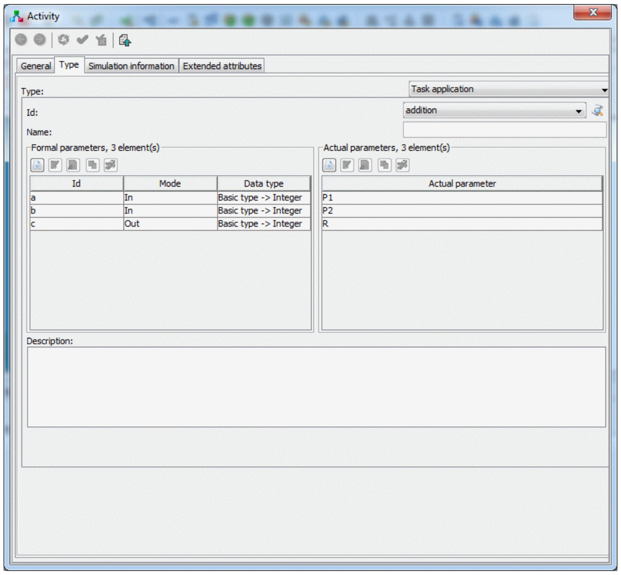
Block activity type property panel contains information about *activity set id*, which is a reference to defined ActivitySet (re-usable block of activities and transitions which shares the context of the workflow process where defined).

A block activity executes referenced ActivitySet or self-contained activities/transitions map. When block activity is being executed, the execution proceeds to the first activity within the ActivitySet it references, and continues within the set until it reaches ActivitySet's exit activity (an activity with no output transitions). Execution then returns to follow the output transitions of the block activity.



Type: Block activity

Activity set Id: Business_Example_Wor1_Ase1



Type: Task application

Id: addition

Name:

Formal parameters, 3 element(s)

Id	Mode	Data type
a	In	Basic type -> Integer
b	In	Basic type -> Integer
c	Out	Basic type -> Integer

Actual parameters, 3 element(s)

Actual parameter
P1
P2
R

Description:

Task-application activity property panel enables to define a reference to the application required for enactment engine to run, in order to perform the activity.

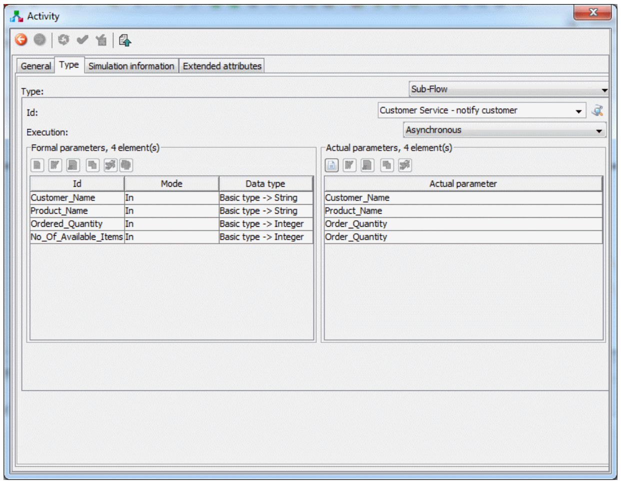
Basically, this tab allows you to pick the desired application by selecting it from the combo-box, and then allows definition of actual parameters (variables or expressions) to pass to the referenced application depending on its formal parameters.

Subflow is type of activity whose implementation is another workflow process.

Firstly, attribute *id* should be set. *Id* defines workflow process that will be executed. In TWE you are setting this attribute by picking the one of the WorkflowProcesses defined within the combo box. Beside the combo, you have a shortcut to display the properties of the WorkflowProcess selected within the combo.

Then, *execution (mode)* attribute should be set. *Synchronous execution mode* suspends execution of calling process until sub-flow is finished. *Asynchronous mode*, spawns a new thread of execution for sub-flow process, which is then executed at its own pace, independently from calling workflow process.

While entering *actual parameters* that will be passed to subflow process, TWE shows the list of corresponding *formal parameters* of the referenced sub-process.



Type: Sub-Flow

Id: Customer Service - notify customer

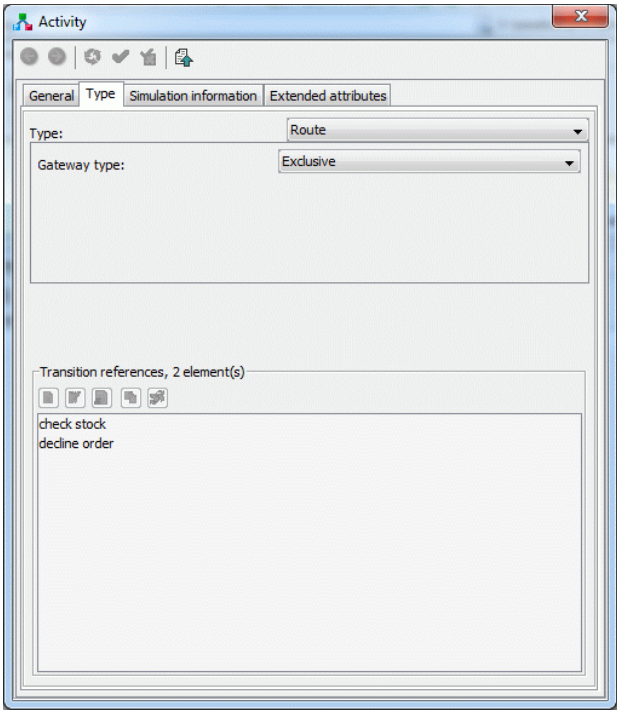
Execution: Asynchronous

Formal parameters, 4 element(s)

Id	Mode	Data type
Customer_Name	In	Basic type -> String
Product_Name	In	Basic type -> String
Ordered_Quantity	In	Basic type -> Integer
No_Of_Available_Items	In	Basic type -> Integer

Actual parameters, 4 element(s)

Actual parameter
Customer_Name
Product_Name
Order_Quantity
Order_Quantity



Type: Route

Gateway type: Exclusive

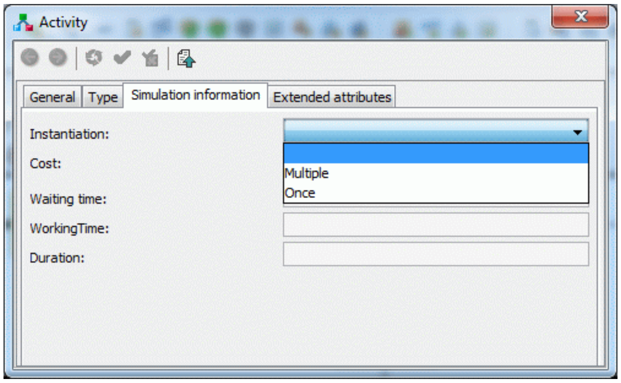
Transition references, 2 element(s)

check stock
decline order

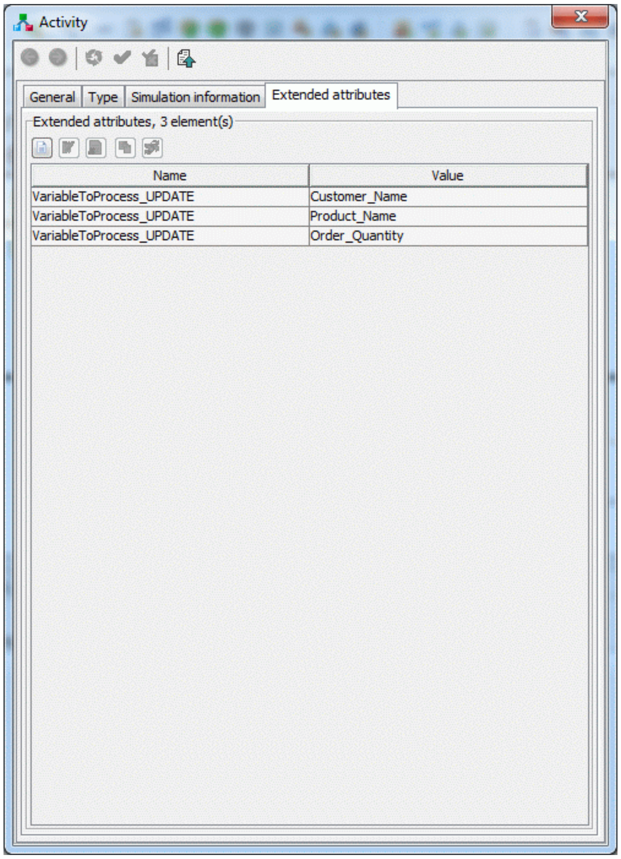
When gateway (routing) activity is selected, type tab allows to change the gateway type from parallel to exclusive or vice-versa. In the case of exclusive gateways with more than one outgoing transition from the gateway, the order of calculating outgoing transition conditions is important. This order is determined in XPDL by the order of TransitionRef elements within its TransitionRefs collection. From here, you can control that order. The way of doing it is to simply change positions of the target activities (the activities that outgoing transitions are leading to) within the given list.

Simulation Information tab

Here are defined various information about simulation: whether the activity is instantiated once or multiple times, cost, waiting and working times and duration.



Extended Attributes tab



Name	Value
VariableToProcess_UPDATE	Customer_Name
VariableToProcess_UPDATE	Product_Name
VariableToProcess_UPDATE	Order_Quantity

It contains information about all extended attributes of the selected activity in form a of a table and operations for their managing. Every table row (extended attribute) is described with *name* and *value*.

Artifacts

To satisfy additional modeling concepts that are not part of the basic set of flow elements (activities and transitions), BPMN provides the concept of Artifacts that can be linked to the existing Activity object through Associations. Thus, Artifacts do **not** affect the basic sequence flow, nor do they affect mappings to execution languages.Artifacts are used to show additional information about a Process that is not directly related to its sequence flow.

At this point, BPMN provides three standard Artifacts: A Data Object, a Group, and an Annotation. TWE currently allows you to use the first two types of artifacts.

An Artifact is a graphical object that provides supporting information about the Process or elements within the Process. However, as mentioned, it does **not** directly affect the flow of the Process. BPMN provides a specific graphical representation for the different artifacts:

Figure 6.4. BPMN Notation for Artifacts

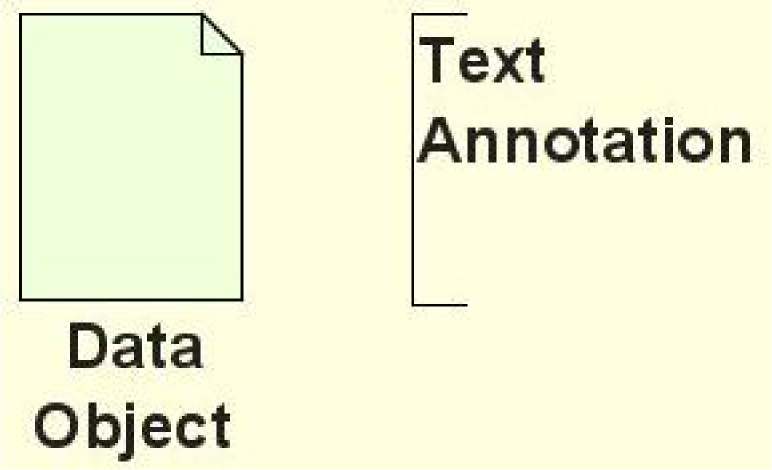


Table 6.19. Artifact attributes

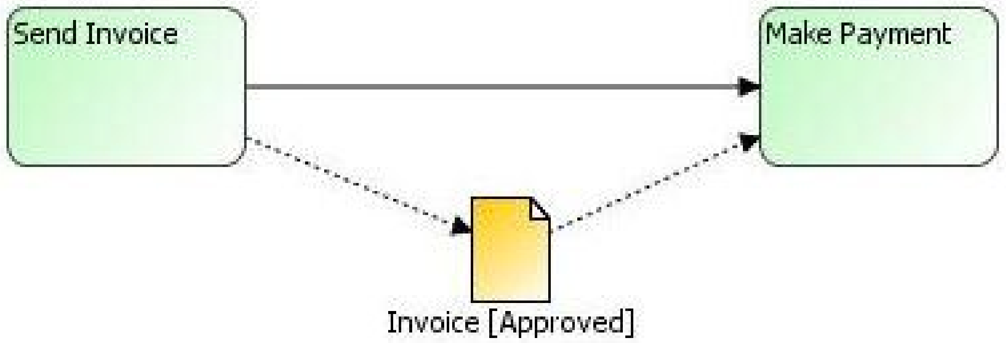
Name	M/O	Description
Name	O	Name of the artifact.
ArtifactType	M	DataObject | Group | Annotation
TextAnnotation	O	Visible textual description.
DataObject	O	When artifact is DataObject type, it describes other attributes of an artifact.

In BPMN, a Data Object is considered an Artifact because it does not have any direct affect on the sequence flow of the Process, but it does provide information about what the Process does. That is, how documents, data, and other objects are used and updated during the Process. While the name - Data Object may imply an electronic document, it can be used to represent many different types of objects, both electronic and physical.

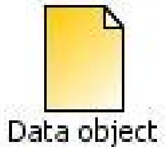
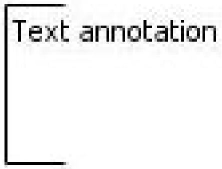
As an Artifact, Data Objects generally will be associated with activities. An Association will be used to make the connection between the Data Object and the Flow Object. This means that the behavior of the Process can be modeled without Data Objects for modelers who want to reduce clutter. The same Process can be modeled with Data Objects for modelers who want to include more information without changing the basic behavior of the Process.

The following is a sample of process modeled with an artifact showing the information about the document flowing from one activity to another:

Figure 6.5. Artifact Sample

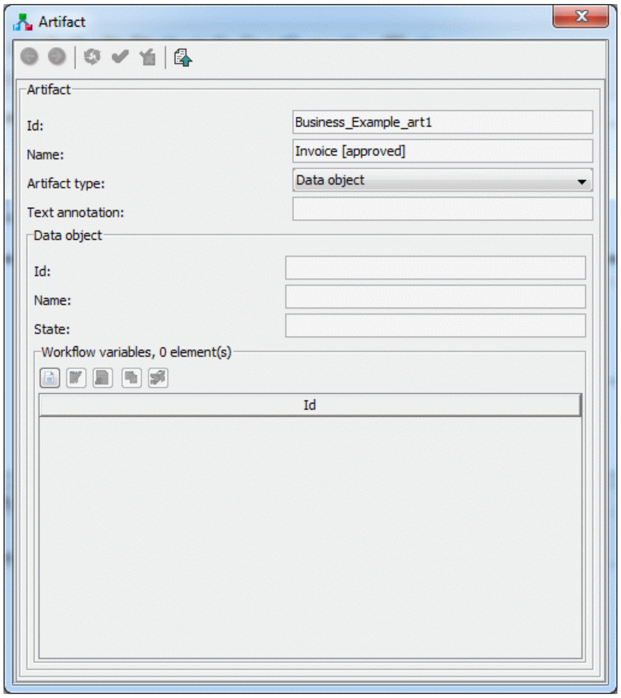


The following table shows the picture of how the graph represents different XPDL artifact type:

	Data object artifact.
	Text annotation artifact.

Icons for inserting different types of artifacts from the graph toolbox are 🗋 for dataobject artifact, and [for text annotation artifact.

The picture on the right shows property panel for defining artifact properties.



Id: Business_Example_art1

Name: Invoice [approved]

Artifact type: Data object

Text annotation:

Id:

Name:

State:

Transitions

Link between two activities is established by transitions. Transitions are more than just the links between activities. They also describe the condition that must be satisfied in order to get from the source to the target activity (This condition is evaluated during workflow execution).

Typically, you will insert the transitions using TWE's Graph component (although it is possible to do it through the property panels).

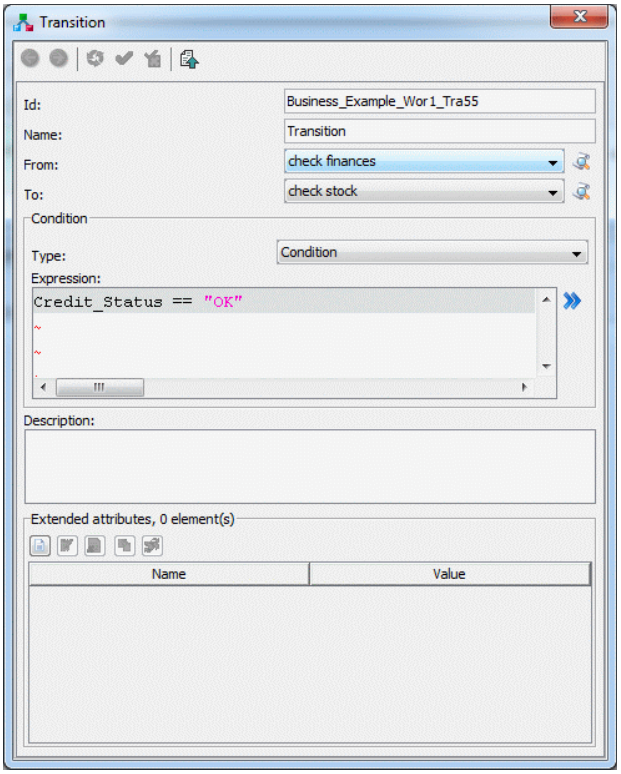
Graph component allows you to chose between conditional →, unconditional →, otherwise →, exception → and default exception → transition for the insertion. If you want to insert a circular transition (a transition from activity to itself), you can chose the type of the transition you like, and double-click on the activity within the graph.

If insertion of the transition is not allowed by some rules (e.g. you can't insert two transitions connecting the same activities), you'll be notified or graph component will simply deny to insert such transition.

Here is the table of transition attributes:

Table 6.20. Transition

Name	**M/O**	**Description**
Condition	O	A Transition condition expression based on workflow relevant data.
Description	O	Short textual description of the Transition.
Extended Attributes	O	Optional extensions to meet individual implementation needs.
Id	M	Used to identify the Transition.
From	M	Determines the FROM source of a Transition (Activity Identifier).
To	M	Determines the TO target of a Transition (Activity Identifier).
Name	O	Text used to identify the Transition.



Id: Business_Example_Wor1_Tra55

Name: Transition

From: check finances

To: check stock

Condition

Type: Condition

Expression:

```
Credit_Status == "OK"
```

Description:

Extended attributes, 0 element(s)

Name	Value

The picture on the left shows a dialog for editing Transition attributes.

TWE helps you when you are composing Transition condition - by clicking the arrow right to the text box for entering condition, you get a list of possible variables you can use within condition. Also, you can change transition from/to attributes by chosing another from the combo-boxes. There is a link next to those combo boxes to navigate to the selected Activity.

Associations

Link between activity and artifact is established by association.

An Association does not have a specific mapping to an execution language element. These objects and the Artifacts they connect to provide additional information for the reader of the BPMN Diagram, but do not directly affect the execution of the Process.

Typically, you will insert the association using TWE's Graph component (although it is possible to do it through the property panels).

Graph component allows you to chose between directional ····✚, non-directional ········ and bi-directional ✚···✚ association for the insertion.

If insertion of the association is not allowed by some rules (e.g. you can't insert two associations connecting the same artifact and activity, or you can't insert association between two artifacts or two activities), graph component will simply deny to insert such association.

Here is the table of transition attributes:

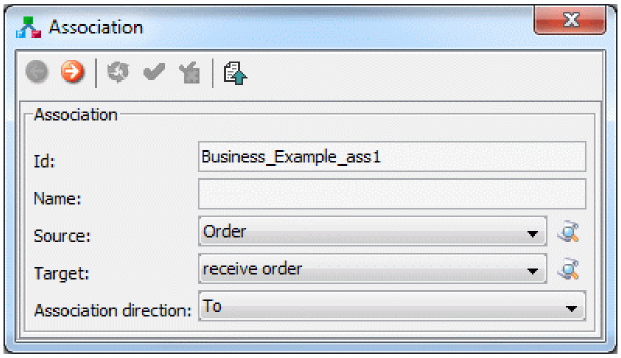
Table 6.21. Association

Name	M/O	Description
Id	M	Used to identify the Transition.
Name	O	Text used to identify association.

Name	M/O	Description
Source	M	Determines the source of an Association (Activity or Artifact Identifier).
Target	M	Determines the target of an Association (Activity or Artifact Identifier).
Association direction	M	The direction for association, possible values: None, To, From, Both

Picture on the right shows a dialog for editing Association attributes.

You can change association source/target attributes by chosing another from the combo-boxes. There is a link next to those combo boxes to navigate to the selected Activity/Artifact.



Association

Id: Business_Example_ass1

Name:

Source: Order

Target: receive order

Association direction: To

Chapter 7. XPDL from Scratch

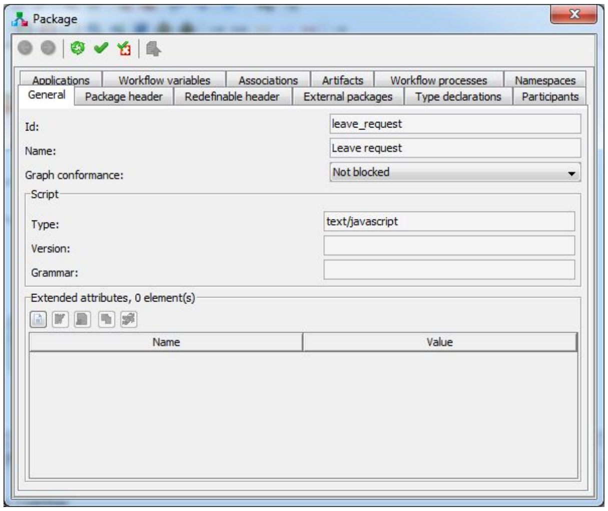
These are the steps that are typically required in order to build new XPDL:

- Click on the icon from the main toolbar (the left-most icon in the toolbar), or select menuitem File->New to create a new Package.

- Click on the icon from the main toolbar or select menuitem Package->Package properties to get the property panel for editing properties of newly created Package:

  Here you can edit some of the package attributes, like Id, Name, the Type of the scripting language used when writing expressions, ...

  Click on the icon from the dialog's toolbar to apply the changes, and to close the dialog



Id: leave_request
Name: Leave request
Graph conformance: Not blocked
Script
Type: text/javascript
Version:
Grammar:
Extended attributes, 0 element(s)

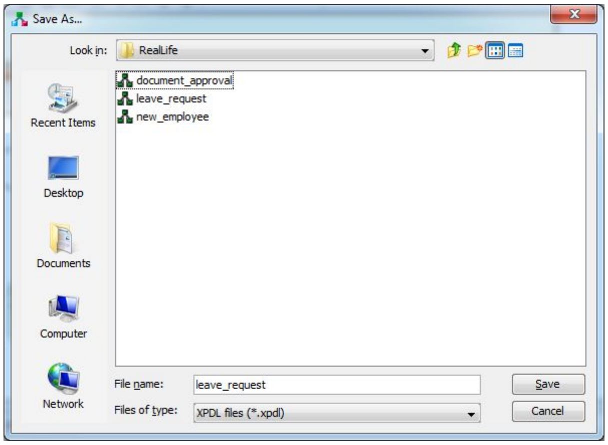
- Click on the icon or from the main toolbar or select menuitem Package->Save or Package->Save As... to save new Package into XPDL file:

  Browse the file system and choose the folder where will you save Package

  Enter the name of the file (the default is the Id of the Package)

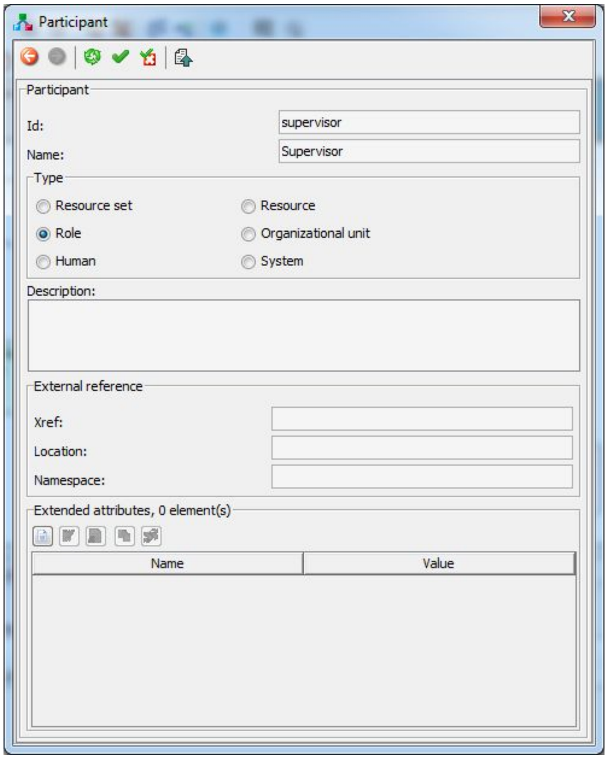
  Select the extension for the XPDL file (the default is xpdl)

  Click Save button



Look in: RealLife
document_approval
leave_request
new_employee
File name: leave_request
Files of type: XPDL files (*.xpdl)

- Click on the icon or select menuitem Package->Participants to get property panel with information about Participants defined on Package level, and then click on the icon on the property panel to define new Participant:



Participant

Id: supervisor

Name: Supervisor

Type

○ Resource set ○ Resource
● Role ○ Organizational unit
○ Human ○ System

Description:

External reference

Xref:

Location:

Namespace:

Extended attributes, 0 element(s)

Name	Value

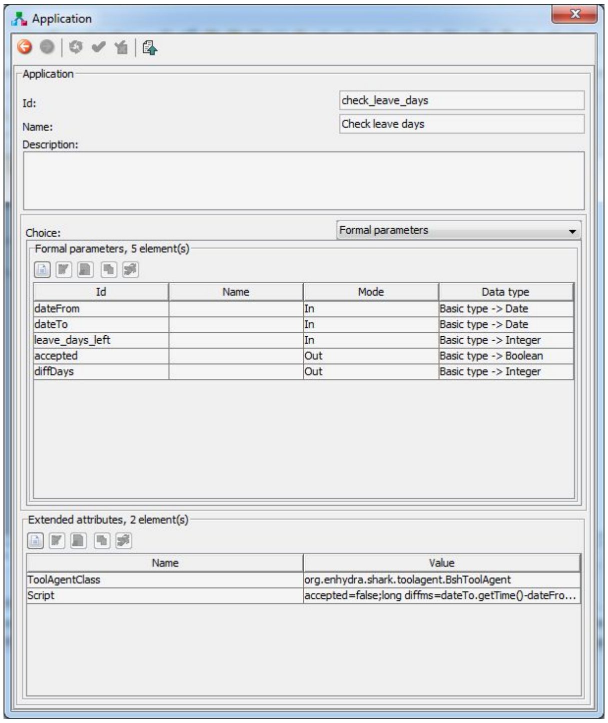
Here you can edit some attributes like Id, Name, Type, Description, ...

Click on the icon ✓ from the dialog's toolbar in order to apply the changes.

Click on the icon ⬅ or ⬆ in order to go back to the previous property panel

Repeat this step until you define all of the participants you need.

- Click on the icon or select menuitem Package->Applications to get property panel with information about Applications defined on Package level, and then click on the icon on the property panel to define new Application:


Application
Application
Choice:
Formal parameters, 5 element(s)
Extended attributes, 2 element(s)

Id: check_leave_days

Name: Check leave days

Description:

Choice: Formal parameters

Id	Name	Mode	Data type
dateFrom		In	Basic type -> Date
dateTo		In	Basic type -> Date
leave_days_left		In	Basic type -> Integer
accepted		Out	Basic type -> Boolean
diffDays		Out	Basic type -> Integer

Name	Value
ToolAgentClass	org.enhydra.shark.toolagent.BshToolAgent
Script	accepted=false;long diffms=dateTo.getTime()-dateFro...

Edit some attributes like Id, Name, Description.

Click on the icon ✓ from the dialog's toolbar in order to apply the changes.

Add necessary Formal parameters by clicking the icon on the Formal parameters section (this will show another property panel for editing newly created Formal parameter - edit it, go back to this property panel, and repeat the step until you define all of them).

If necessary, add Extended attributes for the application by clicking the icon on the Extended attributes section (this will show another property panel for editing newly created Extended attribute - edit it, go back to this property panel, and repeat the step until you define all of them)

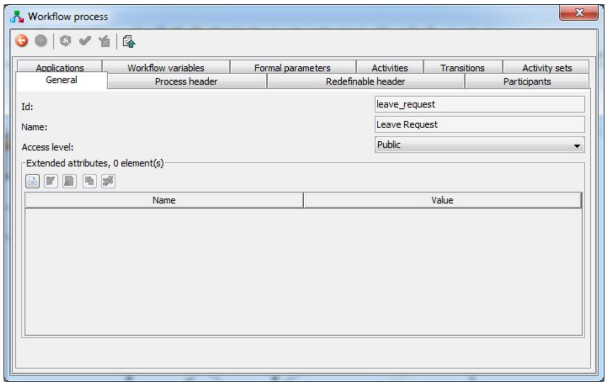
Click on the icon or in order to go back to the previous property panel

Repeat this step until you define all of the applications you need.

After you finish with defining Applications, close the dialog.

- Click on the icon from the main toolbar, or select menuitem Package->Insert new process to create a new WorkflowProcess.

- Click on the icon from the main toolbar or select menuitem Process->Process properties to get the property panel for editing properties of newly created Workflow Process:



Id: leave_request

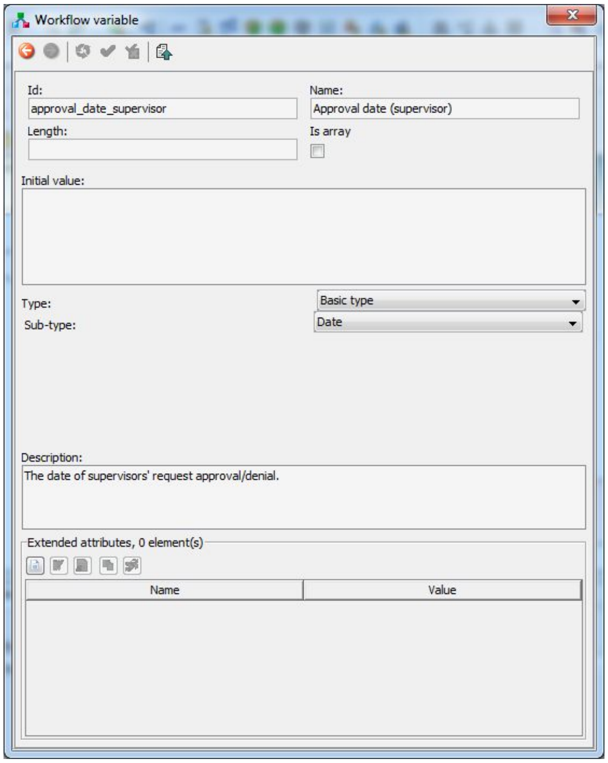
Name: Leave Request

Access level: Public

Here you can edit some of the process attributes, like Id, Name, Access level ...

Click on the icon from the dialog's toolbar to apply the changes, and to close the dialog

- Click on the icon or select menuitem Process->Workflow variables to get property panel with information about Workflow variables defined for the Process, and then click on the icon on the property panel to define new Workflow variable:



Id: approval_date_supervisor

Name: Approval date (supervisor)

Length:

Is array ☐

Initial value:

Type: Basic type

Sub-type: Date

Description: The date of supervisors' request approval/denial.

Extended attributes, 0 element(s)

Name	Value

Here you can edit some attributes like Id, Name, Type, Sub-Type, Initial value, Description, ...

Click on the icon ✓ from the dialog's toolbar in order to apply the changes.

Click on the icon ← or ⇧ in order to go back to the previous property panel

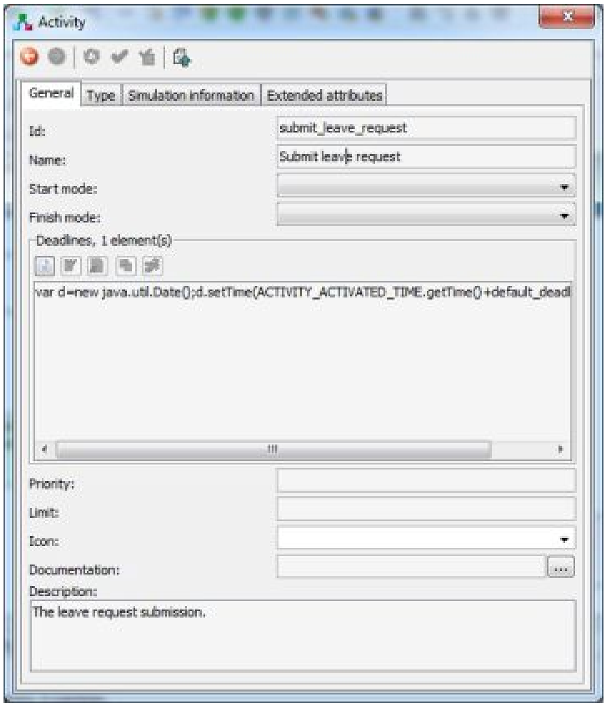
Repeat this step until you define all of the variables you need.

- Click on the icon ⊞ from the graph toolbar. You'll get the list of all Participants you've defined, plus so called Free text expression lane (which is not a representation of the participant from the model but a special Graph swim-lane for holding activities which performer is defined as an expression). By selecting Participant from the list, the lane representing that participant is being inserted into the Graph. Put all the necessary Participant lanes into the Graph (they'll be represented as a swim-lanes).

- Use the icons from the Graph's toolbox (●, ●, ▢, ▣, ▣, ▣, ◈, ⊕, ⟼, ⟶, ⟶, ⟶, ⟶) in order to insert different kinds of activities into the graph, and to connect them.

- Right click on the graph objects representing activities you've inserted to get the popup menu. Select Properties menuitem to get the property panel for editing activity properties (or simply double-click on the activity object in the graph).

  Depending on the activity type, and the number of outgoing transitions in the case of "gateway" (route) activity, you'll get different property panels (different Type tabs).



Id: submit_leave_request

Name: Submit leave request

Start mode:

Finish mode:

Deadlines, 1 element(s)

var d=new java.util.Date();d.setTime(ACTIVITY_ACTIVATED_TIME.getTime()+default_deadl

Priority:

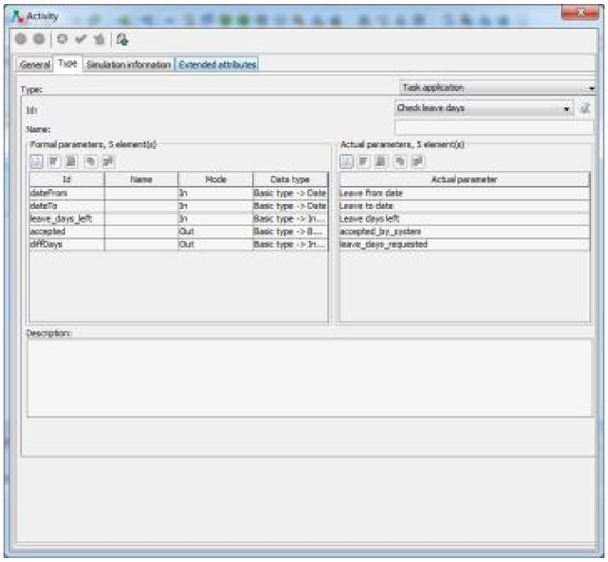
Limit:

Icon:

Documentation:

Description:

The leave request submission.



Type: Task application

Id: Check leave days

Name:

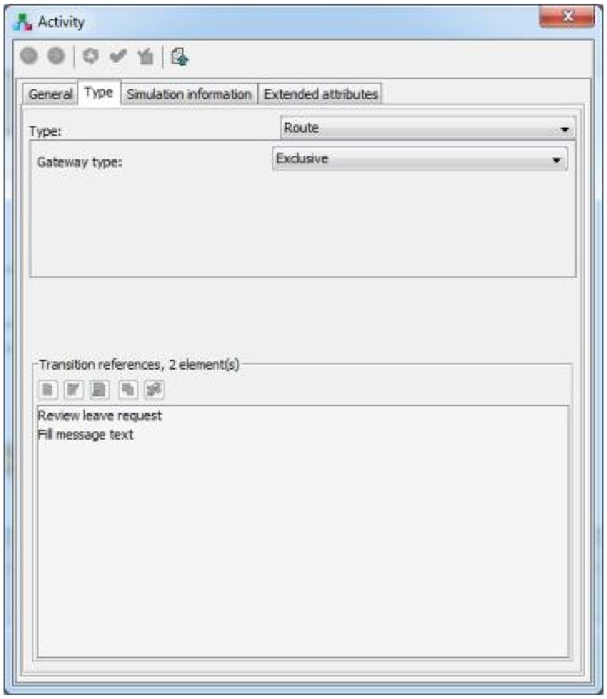
Formal parameters, 5 element(s)

Id	Name	Mode	Data type
dateFrom		In	Basic type -> Date
dateTo		In	Basic type -> Date
leave_days_left		In	Basic type -> In...
accepted		Out	Basic type -> B...
diffDays		Out	Basic type -> In...

Actual parameters, 5 element(s)

Actual parameter
Leave from date
Leave to date
Leave days left
accepted_by_system
leave_days_requested

Description:

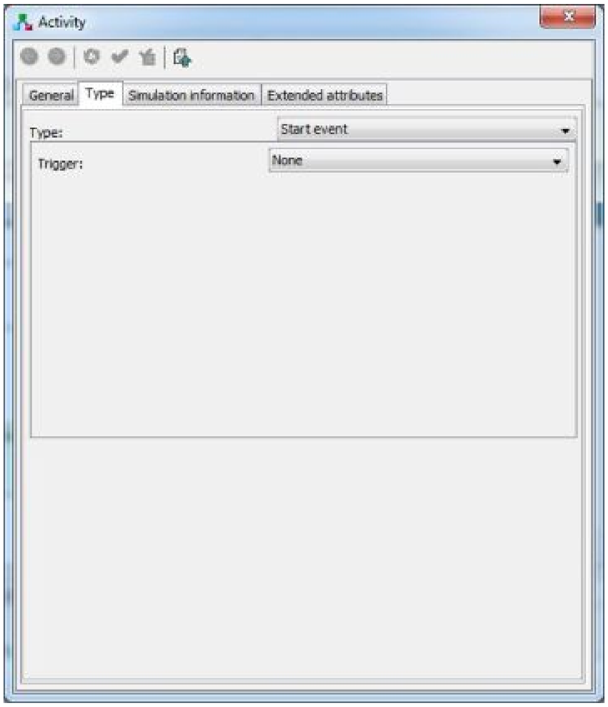


Type: Route

Gateway type: Exclusive

Transition references, 2 element(s)

Review leave request
Fill message text



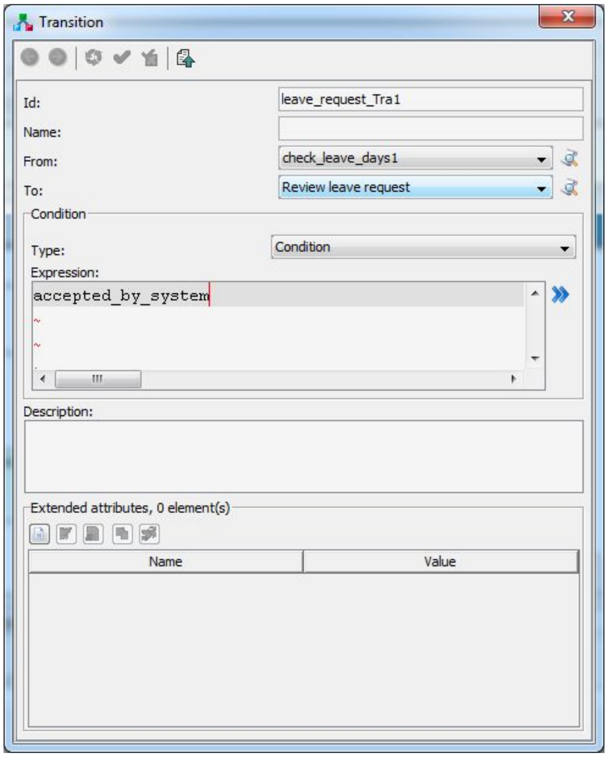
Type: Start event

Trigger: None

Browse through the tabs and edit activity's properties. For the tool activity, add a new Tool and select application reference, and define actual parameters.

Be sure you applied all the changes by clicking the icon ✓ from the dialog's toolbar.

- Right click on the graph objects representing transitions you've inserted to get the popup menu. Select Properties menuitem to get the property panel for editing transition properties (you can also simply double-click on the transition object in the graph):



Id: leave_request_Tra1

Name:

From: check_leave_days1

To: Review leave request

Condition

Type: Condition

Expression:

```
accepted_by_system
```

Description:

Extended attributes, 0 element(s)

Name	Value

If necessary, change transition condition's type.

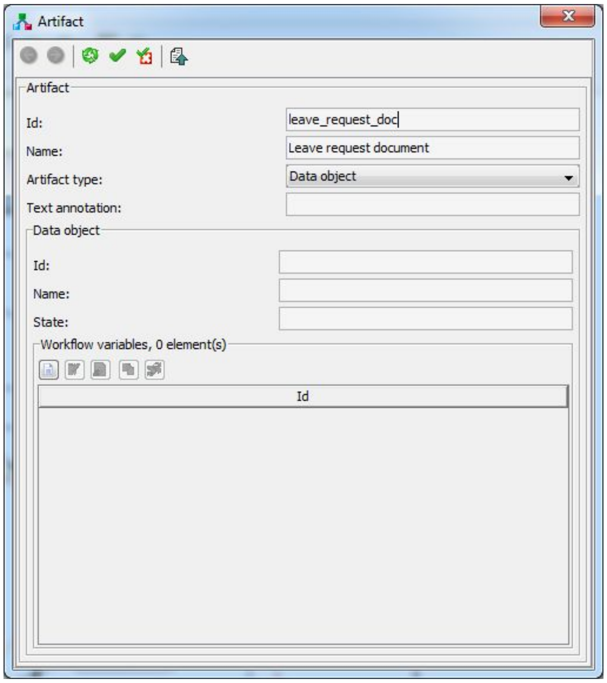
If necessary, define condition expression by the help of the icon next to the text area for defining expression (it'll offer you the list of the variables you can use for the conditions).

Click on the icon ✓ from the dialog's toolbar in order to apply the changes.

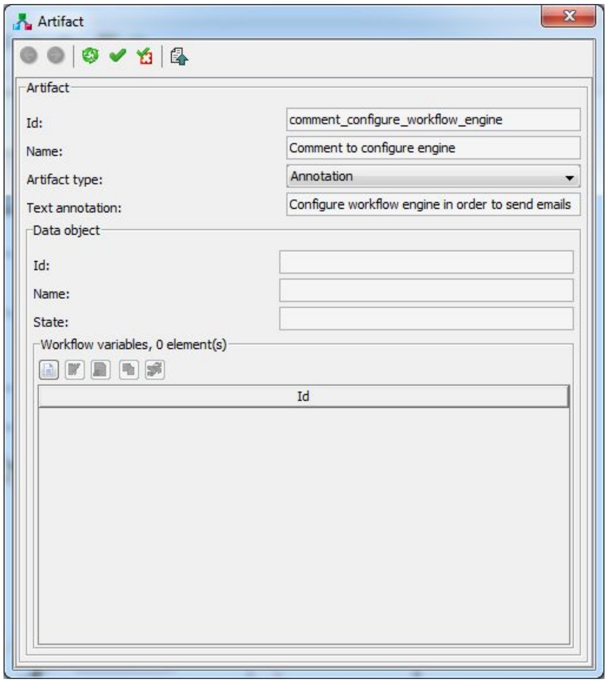
- Use the icons from the Graph's toolbox (🟨, [, ┄┄+, ┄┄┄, +┄┄+) in order to insert different kinds of artifacts into the graph, and to connect them.

- Right click on the graph objects representing artifacts you've inserted to get the popup menu. Select Properties menuitem to get the property panel for editing artifact properties (or simply double-click on the artifact object in the graph).

  Enter meaningful names for data object artifacts, and text annotation for the annotation type artifacts.



Id: leave_request_doc
Name: Leave request document
Artifact type: Data object
Text annotation:



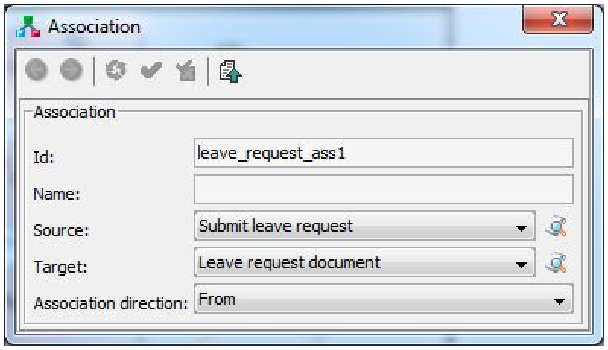
Id: comment_configure_workflow_engine
Name: Comment to configure engine
Artifact type: Annotation
Text annotation: Configure workflow engine in order to send emails

Be sure you applied all the changes by clicking the icon ✓ from the dialog's toolbar.

- If necessary, right click on the graph objects representing associations you've inserted to get the popup menu. Select Properties menuitem to get the property panel for editing transition properties (you can also simply double-click on the association object in the graph):

  If necessary, change association's Id, Name or Direction.

  Click on the icon ✓ from the dialog's toolbar in order to apply the changes.



Id: leave_request_ass1
Name:
Source: Submit leave request
Target: Leave request document
Association direction: From

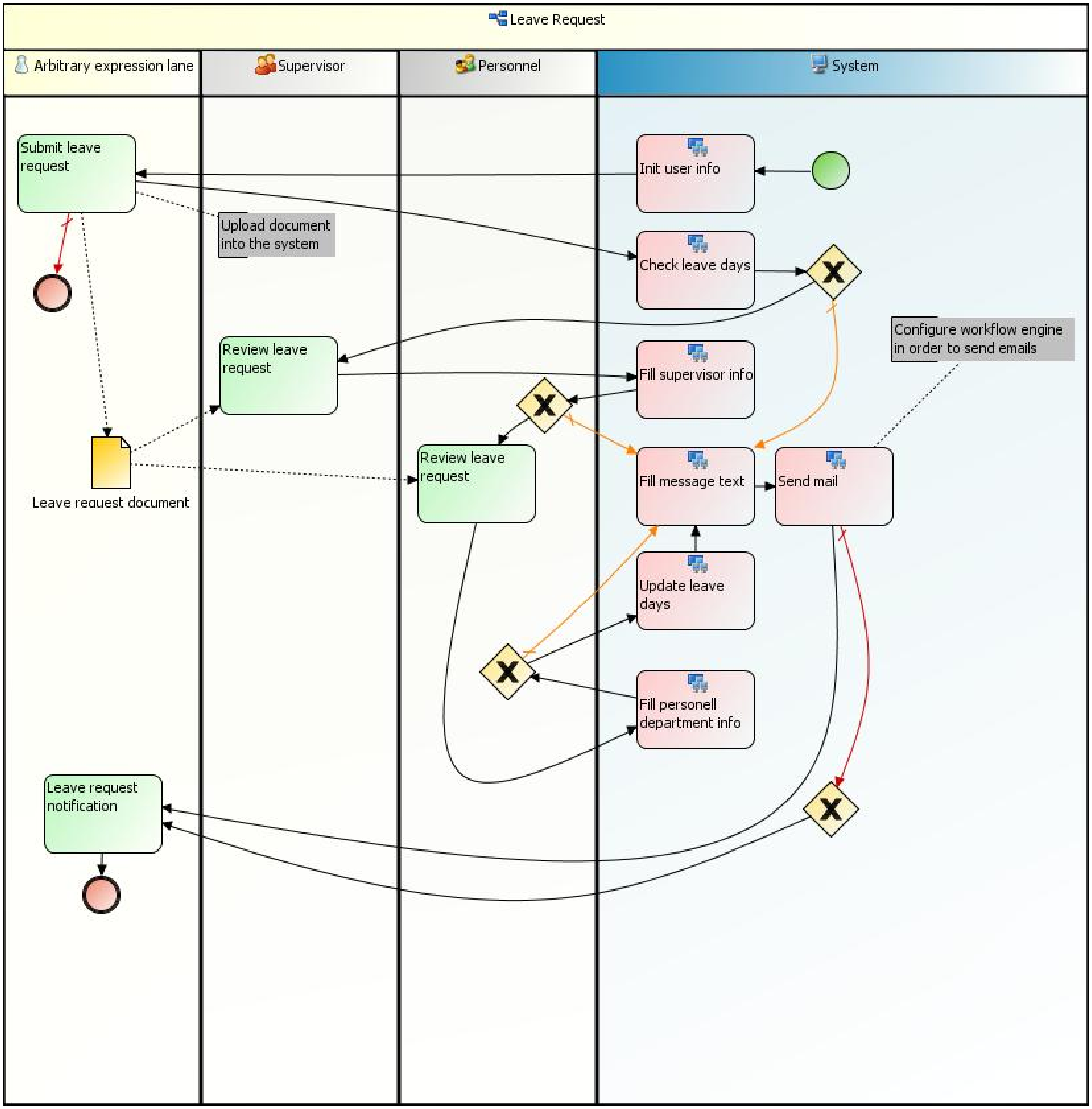
- After you finished with editing, you should check the Package for validity by clicking on the icon ✓ from the main toolbar, or by selecting menuitem Package->Check validity.

  If there are errors or warnings about XPDL model invalidity, the Problems panel will display all the necessary information in order to fix it.

- Do not forget to save the changes you've made: click on the icon 💾 from the main toolbar or select menuitem Package->Save. The changes will be saved in already defined XPDL file.

At the end, the result of the modeling should be something like in the following picture:

Figure 7.1. Leave Request Process



Chapter 8. Configuration

TWE is very configurable in a sense that there are many options that can be changed through TWE's property files.

Almost every component (graph, xpdl view, package tree, ...) has its own property file, but there are also some property files for the TWE's core system components.

The relevant property files of the core system components are:

1. togwebasic.properties
2. componentmanager.properties
3. togwecontroller.properties
4. jawetypes.properties
5. togweeditor.properties
6. transitionhandler.properties
7. xpdlvalidator.properties

Other property files are component specific settings, and these files are:

- infobar.properties
- togwegraphcontroller.properties
- togwexpdlview.properties
- extpkgrelations.properties
- packagenavigator.properties
- detailedpackagenavigator.properties
- propertiespanelcomponent.properties
- searchnavigator.properties
- problemsnavigator.properties
- wfxml.properties
- ldap.properties

The most important of these additional files is probably togwegraphcontroller.properties file. There are settings for graph which is, of course, the most important component.

The properties of non-core components are already explained in a sections explaining components itself, so further text will be related only to configuration of core system components.

NOTE: Most of the settings start with #. This means that they are put under comment and that default settings will be used. In order to change settings, remove # and change settings value

Property file *togwebasic.properties*

This property file contains some basic TWE settings like:

- *Font.Size* and *Font.Name* are used for font settings (default font is Sans Serif, size 11).
- *StartingLocale* sets local settings to be used. Language depends on this settings. The default value is "default", and it uses English language. If you leave this property empty, your system settings will be used, which means if there is a language property file for your system settings, it will be used.
- *LookAndFeelClassName* sets UI manager. Default value for this setting is empty, which means that native UI manager will be used. The possible values for this setting could e.g. be *javax.swing.plaf.metal.MetalLookAndFeel, com.sun.java.swing.plaf.motif.MotifLookAndFeel, ...*
- *UseXPDLPrefix* determines whether to use xpdl name space prefix when saving XPDL file. Default value for this property is "tue".
- *DefaultTransientPackages* contains a list of the absolute paths to XPDL files which will be loaded into TWE as the 'Transient' ones. By default, no transient packages are loaded.
- *FileLocking* - if set to true, TWE will lock open XPDL files. Main XPDL file will be exclusively locked, and external package's XPDL files will have a shared lock

Property file *componentmanager.properties*

This configuration file contains information about which component (Graph, XPDL View, Navigator, ...) will be used in a runtime. It also defines default tab area for the component, and default placement order of the component inside the tab.

To specify that a certain component should be used in a runtime, you must define at least two properties:

1. The first property defines the component to add. E.g. to add Graph component, we would define something like:

   Component.Add.GraphController=org.enhydra.jawe.components.graph.GraphController

   The value of this property is the name of the Java class representing TWE component.

2. The second property defines settings to be used to configure the component. E.g. for the Graph component defined in 1), we would define something like:

   Settings.GraphController = org.enhydra.jawe.components.graph.TogWEGraphControllerSettings

   The value of this property is the name of the Java class representing TWE settings for the given component.

The parameters that end with *.ComponentOrder* are used for setting tab order for each of the TWE frame sections (areas). E.g. to add Graph component to be initially the second one in the 'Main' tab, it should be something like:

Main.ComponentOrder = XPDLViewController GraphController

Note that parameters *UpperStatus* and *LowerStatus* do not have action order, because these areas contain only one component.

Property file *togwecontroller.properties*

This file defines TWE's frame organization (how will it be divided in areas), main menu, tool bar and info bar settings, and TWE's default actions and possibilities. There are a lot of parameters that define these features.

The following parameters can have either value true, or value false:

- *AllowInvalidPackageSaving* - if set to true, saving of invalid packages (XPDL files) is allowed even if the package's *Publication status* property is defined as *Released.*

- *AskOnDeletion* - if set to true, TWE asks for a user confirmation before deleting any element.
- *AskOnDeletionOfReferencedElements* - if set to true, and previous property is set to false, TWE asks for a user confirmation only before deleting elements referenced by other elements.
- *InitialXPDLValidation* - if set to true, XPDL file will be checked for errors at a time it is loaded into TWE
- *DesignTimeValidation* - if set to true, each time after an action that changes XPDL model is performed, the XPDL will be checked for errors.
- *StartMaximized* - if set to true, TWE is started in maximized window.
- *ShowTooltip* - if set to true, tooltips are displayed.

The parameter *UndoHistorySize* should have an integer value, and it defines the number of XPDL model related actions that can be undone/redone. If this number is less than zero, TWE will be configured to have unlimited undo/redo history.

There are also other parameters that can have different string values:

- *Encoding* - defines encoding of files saved with TWE.

  For example:

  ```
  Encoding = UTF-8
  ```

- DoNotAskOnDeletionOfReferencedElementTypes - if the property AskOnDeletion is set to false, and the property AskOnDeletionOfReferencedElements is set to true, the value of this property defines for which referenced elements TWE won't ask for a user confirmation before deleting the element.

  For example:

  ```
  DoNotAskOnDeletionOfReferencedElementTypes = Activity Transition
  ```

The following parameters are defining initial TWE frame settings:

- *FrameSettings* - this parameter defines how will the main application frame be divided. As already explained there are several application areas. Amongst others, there are areas called *main*, *tree*, *special* and *other*. The example of valid value for this paramter is:

  ```
  FrameSettings = V; special H tree; main H other
  ```

  which means that the frame will be first divided vertically. The left part of the frame will be divided horizontally, where *special* area will be in the upper and *tree* area in the lower part, and also the right part will be divided horizontally, where *main* area will be in the upper and *other* area in the lower part.

- *MainDividerLocation* - this property is an integer value that defines the initial location for the main frame divider. In the example above, it is the vertical divider that divides the frame into the left and the right part.
- *FirstSmallDividerLocation* - this property is an integer value that defines the initial location for the first sub-frame divider. In the example above, it is the horizontal divider that divides the left sub-frame into the upper part *special* and the lower part *tree*.
- *SecondSmallDividerLocation* - this property is an integer value that defines the initial location for the second sub-frame divider. In the example above, it is the horizontal divider that divides the right sub-frame into the upper part *main* and the lower part *other*.

There are a lot of parameters that are defining toolbar and menubar content and order, ...

E.g. to change the order of menu items in the File menu, or to leave out some items, you can modify the following:

SubMenu.ActionOrder.File = NewPackage Open Reopen Close - Save SaveAs - @RecentFiles - Exit

Property file *jawetypes.properties*

Here are described elements used in TWE: their types and information about them (what should be their color).

For example, the following section defines the properties for an activity which type is *BlockActivity*

```
#JaWETypes.ActivityType.Id.block = ACTIVITY_BLOCK
#JaWETypes.ActivityType.LangDepName.block = BlockActivityKey
#JaWETypes.ActivityType.Icon.block = org/enhydra/jawe/images/blockactivity.gif
#JaWETypes.ActivityType.Color.block = R=255,G=165,B=121
#JaWETypes.ActivityType.XPDLTemplate.block =
```

Changing the color used to represent this kind of activity will have effect only to the graph component.

There is another advanced feature of TWE which can be utilized through jawetypes.properties file. You can define custom XPDL object types, and even provide the XPDL template fragment to define properties they should have. E.g. if you want to define new type of activity called loop activity, you can have the following configuration:

```
JaWETypes.ActivityType.Id.loop = ACTIVITY_LOOP
JaWETypes.ActivityType.LangDepName.loop = LoopKey
JaWETypes.ActivityType.Icon.loop = org/enhydra/jawe/samples/loopactivity/
images/loopactivity.gif
JaWETypes.ActivityType.Color.loop = R=255,G=155,B=15
JaWETypes.ActivityType.XPDLTemplate.loop = sampleactloop.xml
```

where sampleactloop.xml have to be placed under templates subfloder of the configuration you are using. The content of sampleloop.xml file can be something like:

```
<Activity>
   <BlockActivity BlockId=""/>
   <ExtendedAttributes>
      <ExtendedAttribute Name="LoopCondition" Value=""/>
      <ExtendedAttribute Name="LoopType" Value="While"/>
      <ExtendedAttribute Name="BackToPool" Value="false"/>
      <ExtendedAttribute Name="SetTemporary" Value="false"/>
   </ExtendedAttributes>
</Activity>
```

To define custom language strings, you should edit jawelanguagemisc.properties file in the target configuration folder, and add appropriate [key,value] pairs. For the previous example, you should add a only following [key,value] par:

```
LoopKey = Loop
```

If you need language specific entries for other languages, you just put corresponding property file into the target configuration folder, and define the same property. E.g. you define jawelanguagemisc_de.properties file if you need German translation.

This newly defined activity will automatically appear in the Graph's component toolbox ones you start TWE for this configuration mode.

Property file *togweeditor.properties*

This file defines the properties for the dialog component used to display XPDL element property panels:

- *InlinePanel.ShowModifiedWarning* - if set to true, and if user wants to leave the property panel (by using show previous, show next, show parent element panel actions or actions for closing the dialog) of the element for which he previously changed some properties (within the panel), the user is asked if he wants to save the changes, leave the panel without saving changes or to cancel the action.
- *InlinePanel.DisplayTitle* - if set to true, panel will display element name beneath toolbar.
- *HistorySize* - defines the size of the previous/next panel history. If set to the value less than zero, the history is unlimited.
- *Toolbar.ActionOrder.defaultToolbar* - defines the content and the order of the dialog's toolbar buttons.

There are properties which can be adjusted to fine-tune the basic element property panels L&F, such as alignment, TOP, BOTTOM, RIGHT, LEFT empty space in the panels, the width and height of the text boxes, etc. These are defined by the following set of the properties:

```
XMLBasicPanel.RightAllignment=false
XMLBasicPanel.EmptyBorder.TOP=0
XMLBasicPanel.EmptyBorder.LEFT=3
XMLBasicPanel.EmptyBorder.BOTTOM=4
XMLBasicPanel.EmptyBorder.RIGHT=3
XMLBasicPanel.SimplePanelTextWidth=250
XMLBasicPanel.SimplePanelTextHeight=20
```

You are able to customize so called "group" panels, used to display some complex elements, in order to hide some complex element sub-elements. For example, in order not to display Activity's Id, Deadlines, Priority and Limit, you should set the property *HideSubElements.XMLGroupPanel.Activity* to the following value:

```
HideSubElements.XMLGroupPanel.Activity = Id Deadlines Priority Limit
```

You are able to customize which elements of some collection shouldn't be displayed within so called "table" panels. For example, if you don't want to display extended attributes which name attribute is "SpecEA" or "EASpec", you can define the property:

```
HideElements.XMLTablePanel.ExtendedAttributes.Name = SpecEA EASpec
```

You are able to customize so called "table" panels, used to display some complex element collections, in order to specify which sub-elements will be shown as a table columns. For example, when displaying activities, you can specify to show Activity's Id, Name, Performer, Type, Start mode, Finish mode and Deadlines:

ShowColumns.XMLTablePanel.Activities = Id Name Performer Type StartMode FinishMode Deadlines

There is another customization possible for the so called "combo box" panel. Hence, you can define for which elements the combo box will be disabled (by default nothing is disabled). For example, if you want to disable combo boxes for displaying Activity's Performer and Transition's From and To properties, you should specify the following:

```
XMLComboPanel.DisableCombo = Performer From To
```

As with all other components, you are also able to define the action order inside a toolbar.

Property file *transitionhandler.properties*

This file enables limitation of number of incoming or outgoing transition for each specific activity type.

For block activity the following parameters are important:

- Transitions.moreOutgoing.Activity.Type.ACTIVITY_BLOCK

If the value of this parameter is set to true, the block activity can have more than one outgoing transition. If set to false, it can have only one outgoing transition.

- Transitions.moreIncoming.Activity.Type.ACTIVITY_BLOCK

  If the value of this parameter is set to true, the block activity can have more than one incoming transition. If set to false, it can have only one incoming transition.

For generic activity the following parameters are important:

- Transitions.moreOutgoing.Activity.Type.ACTIVITY_NO

  If the value of this parameter is set to true, the No implementation activity can have more than one outgoing transition. If set to false, it can have only one outgoing transition.

- Transitions.moreIncoming.Activity.Type.ACTIVITY_NO

  If the value of this parameter is set to true, the No implementation activity can have more than one incoming transition. If set to false, it can have only one incoming transition.

For route activity the following parameters are important:

- Transitions.moreOutgoing.Activity.Type.ACTIVITY_ROUTE

  If the value of this parameter is set to true, the route activity can have more than one outgoing transition. If set to false, tit can have only one outgoing transition.

- Transitions.moreIncoming.Activity.Type.ACTIVITY_ROUTE

  If the value of this parameter is set to true, the route activity can have more than one incoming transition. If set to false, it can have only one incoming transition.

For sub-flow activity the following parameters are important:

- Transitions.moreOutgoing.Activity.Type.ACTIVITY_SUBFLOW

  If the value of this parameter is set to true, the sub-flow activity can have more than one outgoing transition. If set to false, it can have only one outgoing transition.

- Transitions.moreIncoming.Activity.Type.ACTIVITY_SUBFLOW

  If the value of this parameter is set to true, the sub-flow activity can have more than one incoming transition. If set to false, it can have only one incoming transition.

For tools activity the following parameters are important:

- Transitions.moreOutgoing.Activity.Type.ACTIVITY_TOOL

  If the value of this parameter is set to true, the tool activity can have more than one outgoing transition. If set to false, it can have only one outgoing transition.

- Transitions.moreIncoming.Activity.Type.ACTIVITY_TOOL

  If the value of this parameter is set to true, the tool activity can have more than one incoming transition. If set to false, it can have only one incoming transition.

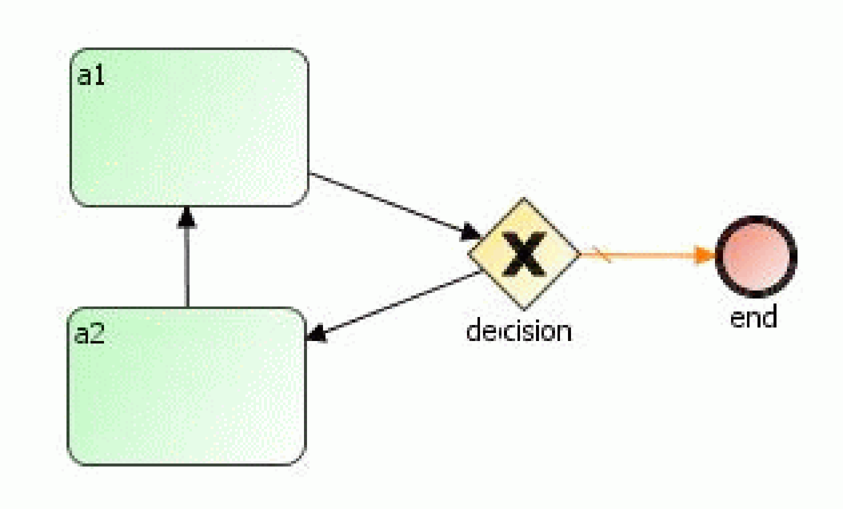
The XPDL validator will validate XPDLs also according to the previously mentioned properties.

Property file *xpdlvalidator.properties*

This file contains several settings that determine some details about validation of XPDL files:

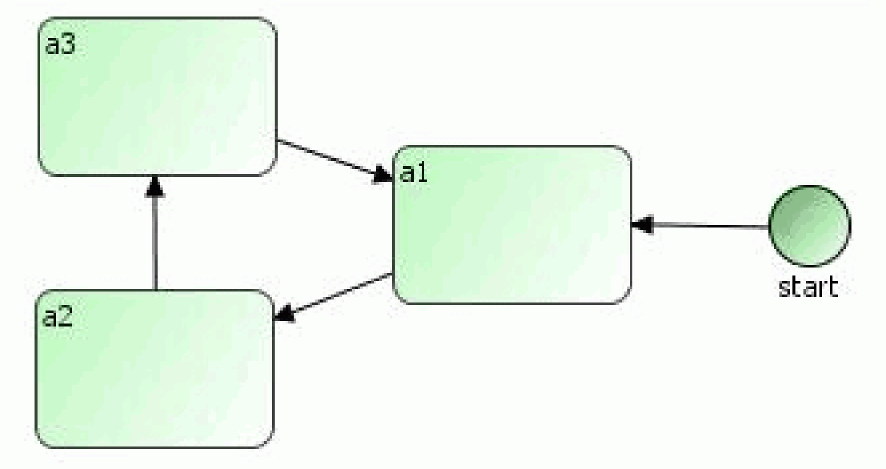
- *AllowUndefinedStart* - if set to false, TWE will require that process and activity set graphs must have at least one starting activity. Otherwise, you will be able to create graphs without starting points, and TWE will not complain. You may have noticed that TWE also considers activity with only circular incoming transition to be the starting point of the graph, so this regards only to graph structures similar to the one bellow:

Figure 8.1. Undefined start example



- *AllowUndefinedEnd* - if set to false, TWE will require that process and activity set graphs must have at least one ending activity. Otherwise, you will be able to create graphs without ending points, and TWE will not complain. You may have noticed that TWE also considers activity with only circular outgoing transition to be the ending point of the graph, so this regards only to graph structures similar to the one bellow:

Figure 8.2. Undefined end example



- *ValidateSubFlowReferences* - if set to true, TWE will consider an error if SubFlow activity does not reference to the valid process from the XPDL or one of the XPDLs referenced through external package concept.

If you know that you will create XPDLs where you'll use SubFlow activities to reference a remote process definition, you should set this property to false, in order not to have validation errors reported.

- *ValidatePerformerExpressions* - if set to true, TWE will consider a warning if activity's performer is not a reference to an existing Participant and it's expression does not look like valid performer expression.
- *ValidateActualParameterExpressions* - if set to true, TWE will consider a warning if actual parameter is not a reference to an existing variable (DataField or FormalParameter) and it's expression does not look like valid actual parameter expression.
- *ValidateConditionExpressions* - if set to true, TWE will consider a warning if transition's condition does not look like valid expression.
- *ValidateDeadlineExpressions* - if set to true, TWE will consider a warning if deadline condition does not look like valid expression.
- *ValidateUnusedVariables* - if set to true, TWE will consider a warning if XPDL variable (DataField or WorkflowProcess's FormalParameter) are not used anywhere (as actual parameter or inside expression).
- *ValidateConditionByType* - if set to true, TWE will generate a warning in following cases:
  - Condition type is not defined, and there is condition expression defined
  - Condition type is 'DEFAULT_EXCEPTION' and there is condition expression defined
  - Condition type is 'OTHERWISE' and there is condition expression defined
  - Condition type is 'EXCEPTION' and there is no condition expression defined
  - Condition type is 'CONDITION' and there is no condition expression defined

Chapter 9. Customization

Need for customization

In order to make the life easier for the XPDL developers, and to make developing of XPDLs less error prone, it could be a good idea to customize TWE. One way of customization can be introduction of new activity types (e.g. automated activities for sending e-mails with customized property panels for entering all the information, a special kind of deadline element, a special XPDL template for creating a new workflow process, ...).

Thanks to its internal architecture, TWE can be easily customized in order to support these requirements.

Example - activity for sending an e-mail

To illustrate previous statement, let's say that in all of our processes, we are intensivelly using Tool activity for sending e-mails.

If we use standard TWE implementation, everytime we want to insert such activity in the process (e.g. via Graph component), we first need to insert a standard Tool activitiy using TWE's Graph component toolbox. After that, we need to open the property panel for the activity, go to the Type section, and create a new Tool for this activity. When new Tool is created, we need to edit it, where we need to select a reference to the Application definition for sending e-mails, and than to create as many ActualParameters for the Tool as there are FormalParameters for the Send E-Mail Application definition. And finally, we need to enter appropriate values for this ActualParameters (either a references to the variables, or some text expressions).

You have to admit that there is a lot of work to do, and it can be quite anoying if you have to do it many times. Also, it is error prone.

What we can do is to extend standard TWE functionallity, and to introduce a new type of an activity for sending e-mails, which will be the customization of the standard Tool activity. We can offer a new button in Graph's toolbar for creating such kind of an activity. This button would have a special icon, and e-mail-ing activity when inserted into graph would also have this icon to be diferentiated amongst others. Also, when such activity is created, its XPDL content would be automatically pre-filled, so it would always have a Tool which would reference the Application for sending an e-mails, and this Tool would already have a default values for all necessary ActualParameters. But this is not all, we can also offer a special property panel for entering e-mail parameters (ActualParameters) - e.g, we can have a combo-box with a list of e-mail addresses (which could be filled through the LDAP), we can have a text box for entering mail subject, and a text area for entering mail content.

Available customizations

TWE distribution comes with several customized configurations. You can switch from one configuration to another one if you select Settings->Configuration and select desired configuration. TWE will be re-configured to use selected configuration.

Available configurations are:

- default - default TWE configuration.
- shark - configuration specific to TWE usage with Shark engine. Includes special property panels for editing some of extended attributes, shark specific restrictions and validation according to these restrictions.
- samples-loopactivity - sample configuration that introduces new type of activity along with the special icon for this activity and special property panel for editing its properties.

Chapter 10. Extended Attribute Reference

TWE uses XPDL ability to store some additional information through extended attributes. TWE might add special extended attributes to package, external package, workflow process, activity and transition elements.

Extended attribute that can be added to any custom main XPDL object (Activity, Application,DataField, Transition, WorkflowProcess, Package, etc.) is:

- JaWE_TYPE - it specifies the custom type Id of the element

Extended attributes added for package element are:

- EDITING_TOOL - used to mark that XPDL was edited by TWE
- EDITING_TOOL_VERSION - stores information about TWE version
- JaWE_CONFIGURATION - stores information about TWE configuration used to create XPDL

In the versions before 4.x, before we switched to XPDL 2 specification, there were extended attributes that were used by TWE's Graph component in order to hold the information about the graphical properties of elements, such as position, orientation, order, ...

These attributes are not used by TWE any more, since graphical information is now stored in corresponding XPDL 2 specification elements.

However, these attributes are important when TWE 4.x opens TWE 3.x or older XPDL files, and affect the conversion from XPDL 1 to XPDL 2.

Here is a list of those "old" extended attributes:

The ones that were added to workflow processes element are:

- JaWE_GRAPH_BLOCK_PARTICIPANT_ORIENTATION - information about orientation of participants (swim-lanes) in activity set's graph
- JaWE_GRAPH_WORKFLOW_PARTICIPANT_ORIENTATION - information about orientation of participants (swim-lanes) in worfklow process's graph
- JaWE_GRAPH_BLOCK_PARTICIPANT_ORDER - holds information about participant order for activity sets in this process
- JaWE_GRAPH_WORKFLOW_PARTICIPANT_ORDER - holds information about participant order for workflow process
- JaWE_GRAPH_START_OF_BLOCK - information about start bubble inside activity set
- JaWE_GRAPH_END_OF_BLOCK - information about end bubble inside activity set
- JaWE_GRAPH_START_OF_WORKFLOW - information about start bubble inside workflow process
- JaWE_GRAPH_END_OF_WORKFLOW - information about end bubble inside workflow process

The ones that were added to activities are:

- JaWE_GRAPH_PARTICIPANT_ID - holds information about swim-lane (representation of the XPDL Participant, or some special kind of swim-lanes for defining activity performer as an expression) that holds activity.
- JaWE_GRAPH_OFFSET - information about location of activity

The ones that were added to transitions are:

- JaWE_GRAPH_TRANSITION_STYLE - information about drawing style for a transition
- JaWE_GRAPH_BREAK_POINTS - information about break point offsets of a transition

Chapter 11. Release Notes

Release 4.4-1

- Updated TXM to version 1.3-1
- Updated commons-io library to version 2.1
- Updated itext5 to version 5.13
-
-
- Updated DOCBOOK DTD to version 5.0
- Updated 3-line titlepage of documentation (PDF)
- Project structure changed according to the new standard (no dependency output anymore, components output instead).
- Linux builds improved
- NSIS and exe installation improvement
- Shark mode: Reporting an error if Name fields of WorkflowProcess/Activity, Id fields of WorkflowProcess/Activity or ExceptionName field length is inappropriate
- Allowing saving of XPDLs with validation warnings in "Released" XPDL Publication Status
- WfXML: URL sample displayed
- Icon and Title now appears in Help->Manual dialog
- Java7 compatibility: - refactored getType/setType methods of JaWEComponent interface to getComponentType/setComponentType - replaced propriatary SUN classes for JPEG handling (from com.sun.image.codec.jpeg package) with ImageIO API (SaveAsJPG.java)

Release 4.3-1

- LDAPPageSize option added to the configuration - now searches will not return partial results because of LDAP server's size limit

Release 4.2-1

- Code modified to support WfXML access through HTTPS
- Now only producing twe.zip for TWS dependency
- Build process changed (preparations for new TAB based release):
    - now possible to manualy specify "buildid" configuration parameter (e.g. configure -buildid 20110721-0808)

  - build scripts corrected so the sources can be built if they are in the folder which contains spaces in the path
  - try/catch blocks removed from documentation build.xml file
  - components folder in the distribution output/components files in the sources (TAB support)
- TXM library updated to version 1.2-1

Release 4.1-1

- Apache Ant libraries updated to version 1.8.2 (ant.jar and ant-launcher.jar)
  - Removed nodeps.jar
- Xerces library updated to version 2.11.0 (xercesImpl.jar).
- Apache Common I/O library updated to version 2.0.1 (commons-io.jar).
- XML Commons library updated to version 1.4_01 (xml-apis.jar).
- Xml Graphics Commons library updated to version 1.4 (xmlgraphics-commons.jar).
- FOP library updated to version 1.0 (fop.jar).
- iText library upgraded to version 5.0.6 (itext.jar).
- DocBook-XSL libraries updated to version 1.76.1 (docbook-xsl).
- Added fix for "Save as PDF" bug.
- Added support for silent installation.
- Updated contact email to `<twe@togetherteam.co.th.>`
- Updated build script to support building of debug version and added "debug" folder to output.
- Removed Java API documentation from distribution zip/ tar.gz/ exe/ rpm files.
- Added fix for bug with copying processes/ activity sets from "transient" packages.
- Added fix: not creating unnecessary NodeGraphicsInfo element for Pools when process/activity set gets copied.
- Updated JAR's MANIFEST information
- Added Together Read Registry (TRR) library version 1.0-1
  - Removed readregistry.nsi
- Standardized the project sources:
  - Updated directory names, filenames and output directory structure.
- Removed Together Application Server (TAS) dependency target.
- Added support for "Script" Activity type (TaskScript from XPDL).
- Updated to improve property panels for DataField and FormalParameter elements.

- Updated property panels for ActualParameter/InitialValue/DeadlineDuration/Condition elements
- Added various improvements.

Release 4.0-1

- Now supporting XPDL 2.1 and BPMN
- Using jxpdl.jar file, the output of Together XPDL Model (TXM)/jXPDL project. This contains the XPDL 2 model classes that were previously the part of Enhydra Shark (Together Workflow Server) project.
- Automatic migration of XPDL 1.0 files into XPDL 2.1 files:
  - Inserting pools and lanes for each workflow process definition
  - Migrating Ids for activities and activity sets - must be unique on the package level (done by TXM)
  - Migrating Tool activities into Task-Application activities, if there are more than one tool for the activity - new activities get created and connected sequentially (done by TXM)
  - Migrating activities (other than Route activities) with Join type different than XOR and Split type different than AND by creating additional Route (gateway) activities that are containing those Join/Split types, and connecting them sequentially (done by TXM)
  - Migrating Route activities with different Join/Split type into two seperate activities and connecting them sequentially (done by TXM)
  - Migrating old Deadline's DeadlineCondition into DeadlineDuration sub-element, according to new schema (done by TXM)
  - Migrating old XOR/AND Join/Split types into Exclusive/Parallel, according to new schema (done by TXM)
  - Removing FormalParameter Index attribute, according to new schema (done by TXM)
  - Migrating IsArray attribute value (of DataField element) from TRUE into true and from FALSE into false, according to new schema (done by TXM)
  - Migrating Activity's Start/Finish mode elements into appropriate attributes according to new schema (done by TXM)
  - Migrating Activity's Performer element into Performers element according to new schema (done by TXM)
  - Migrating order of WorkflowProcess sub-elements (DataFields,Participants,Applications -> Participants, Applications, DataFields) (done by TXM)
- Using BPMN graphical notation for Graph component
- Partial XPDL2.1/BPMN support:
  - Using Pools and Lanes (supported nesting of lanes)
  - Support for Artifacts (DataObject and Annotation type) and Associations
  - Support for Start and End event Activities
  - Support for Task-Application activities (Tool activities from XPDL 2.1 converted into Task-Application activities)

  - Support for Graphical notation elements that hold information about position, color, etc. of the objects (removed extended attributes previously used for that purpose)
- Introduced possibility to change graphical element colors and sizes (information written into appropriate XPDL2.1 entities)
- Sub-Flow and Block activity now showing their graphs when single-clicking on the rectangle area that graphically describes such activities
- New actions to show and hide transition conditions (expressions) in/from the graph
- New actions to show/hide artifacts and associations in/from the graph
- Property panels for elements adjusted according to new XPDL 2.1 schema/specification
- Validation adjusted according to new XPDL 2.1 schema/specification
- Documentation updated
- XPDL2.1 samples added (now there are old XPDL1.0 samples,and corresponding XPDL2.1 samples)
- Build procedure updated
- twe-includes.xlsx file with the list of 3rd party libraries updated
- Removed wfmopen configuration since WfMOpen engine still does not support XPDL 2
- Removed purexpdl configuration - does not make sense any more since TWE 4.x does not use extended attributes for storing graphical information (standard XPDL 2 elements are used now)

Release 3.3-1

- Updated sharkxpdlmodel.jar to version 3.3-1
- Docbook upgraded to the version 1.75-2, removed enhydra specific stuff
- Xalan 2.7-1 added to the project and used instead of saxon to generate docbook docu
- jEdit-Syntax binary code replaced with "home-build" version of projects' CVS sources. (The problem was that the downloaded 2.2.1 version source code didn't match the binaries) Several *.java source files updated to the new package names
- Saxon removed from the project
- batik-xxx.jar files removed from the project (using batik.jar comming with FOP) - build scripts modified
- Added new XPDL (real life) samples
- Release notes merged into the single docbook documentation file
- Fix: issues with defining Record and Union type variables
- Link to homepage changed to the new address
- Company name in various source files changed to Together Teamsolutions Co., Ltd.

- Source distribution does not contain unnecessary temporary files and build.properties file anymore
- Added copyright and GPL V3 comment at top of every source file where missing (including *.xml, *.properties, *.xpdl... files)
- BuildID.txt file added to the binary output - it specifies the time when the release was built
- Distribution packaging changed:
    - twe-doc-x.y-z.zip and twe-screen-x.y-z.zip now also end up into community folder
    - Webstart distribution now packed into twe-x.y-z.webstart.zip file
    - Documentation in documentation folder of the distribution is now unpacked
    - twe-doc-current.pdf with the same content as the current version of project docu ends up into community folder
    - BuildID.txt file ends up into internal folder
    - Added TAS dependency build (the same as originals but with the name not including version and release, and with both Windows and Linux scripts included)
    - Added twe.zip and twe.tar.gz to TWS dependency build (the same as originals but with the name not including version and release)
- Docbook documentation file twe.xml renamed to twe-doc.xml
- Docbook docu updated
    - Documentation content updated
    - Added section into documentation Build guide about all possible configure/make targets
    - Build guide updated with the part related to sign.properties
    - Changed license to FDL version 1.3
    - Fixed docbook docu and build scripts to solve picture and table sizing issues in PDFs
    - Fixed issues with programlisting elements (taking care that lines are not too long)
- Program group entry for API docu removed
- XPDL sample files opened and saved with the newest version of editor
- Standardized make/configure targets
- Improved make/configure scripts for windows and linux
- Improved build procedure so it doesn't fail if sign.properties file does not contain right information
- improved NSIS script - now size of TWE is displayed in Control panel Add/Remove
- twe-includes.xlsx file with the list of 3rd party libraries updated.
- Branding context and build procedure updated
- Removed -optimized parameter for configure script

- Project license now goes to licenses folder, together with other licenses
- Fixed issue with building RPM distributions from SVN sources on Linux
- Build number (in about box) does not contain C anymore
- When XPDL file gets created by the editor, Vendor name now set to (c) Together Teamsolutions Co., Ltd.

Release 3.2-2

- Added copyright and GPL V3 comment at top of every source file
- Updated sharkxpdlmodel.jar to version 3.2-2
- Fix in Utils.java class - path with empty space problem and findPropertyFiles() method usage in eclipse

Release 3.2-1

- code restructured
- updated base component's JAR files to more recent versions (batik, ant, itext, xerces, saxon, fop, java help, jedit syntax and JGraph). Code modified accordingly.
- updated build scripts (build procedure changes)
- new splash screen
- project moved to source forge

Release 3.1-3

- updated build scripts
- sharkxpdlmodel.jar version 3.1-3 included

Release 3.1-2

- updated build scripts

Release 3.1-1

- updated build scripts
- minor changes

Release 3.0-2

- introduced possibility to start JaWE in non-frame/dialog mode
- NSIS tool is updated to version 2.46

- readregistry.nsi script is improved - readregistry.exe is able to get information about Java from 64bit part of the registry.

Release 3.0-1

- Moved completely to open-source, with license changed to GPL V3
- All the sources available on OW2 including the ones for branding TWE
- Included missing features of JPEd (editor based on previous community version of TWE):
  - No annoying 30 seconds splash screen

    since now community version of TWE has all the fatures professional version had, and does not display annoying splash screen

  - Do not lose layout during load/save operations:

    this version of TWE by default doesn't lose layouting information, there is a special "configuration" called "Pure XPDL" where all TWE specific extended attributes are not persisted into XPDL, including the ones that cause lost of layouting information during load/save)

  - The Plugins interface allows you to write wizards, custom XPDL elements editors, custom tooltips, custom views. With it you can easily make jped company requirements

    TWE architecture is pluggable, thus one can write any kind of plug-ins, can provide custom XPDL element editors (additional ones comparing to JPEd and previous TWE community version exist in this new version), can configure different tooltip generator etc...

  - Generate PDF reports of your workflow. Get a complete view of your workflow in a ready to print format. Plugins can be used also to customize PDF outputs.

    The part of the code is taken from JPEd and included in TWE

  - A fully customizable interface, using properties file and system properties: - Restrict the availabe actions in interface or to extend them at glance, depending your target users. - Easily create a custom layout that is easier to manipulate for end-users. - Combined with a wizard plugin, you can prefill an XPDL or auto update it, letting user concentrate only on you core buisness.

    TWE has ability to configure user interface through property files, to restrict/extend available actions or to define which parts of XPDL to hide to the user, define layouts, and also by custom core plug-ins.

    It has a powerful templating mechanism, where one can easily add a new type of prefilled XPDL object, and in the case of activities to easily include it in the toolbox just by configuration and property files

  - Generate real vectorial (SVG) output of graph, including the Ids of activity, participants and transitions. (Zoom at any level with any SVG enabled tool or use it in a workflow engine web interface to highlight current status with just a simple svg stylesheet.)

    This part of code was taken from JPEd and included in TWE

  - Quick search dropdowns make it easy to handle long list of participants

    Already existed in TWE community

  - For shark users, an integrated form editor which also add additional informations in tooltips and PDF

  TWE has a special shark configuration, where not only this feature is provided, but also customized panels adjusted to Shark's extended attributes and other options, special Shark XPDL validation, etc.

  The same thing is done for WfMOpen (special WfMOpen configuration)

  - See the XPDL fragment of each element you are editing, know exactly what you are doing at XPDL level in each dialog.

    In TWE, XPDL View shows the fragment of any selected element in XPDL

  - Condition editor, gives you a dropdown box to select workflow variables and provides syntax highlighting for scripted conditions

    This feature existed in TWE professional version and thus appears in this version now.

  - Customize JPEd logo to fit your company need, provide a splash logo when editor is loading, using a simple System property

    TWE sources now have a "branding" capability. You can brand the whole project (icons, setup, splash logo, about box information) and get setup.exe file for installing such product

- XPDL Model code moved to Shark (LGPL licensed) and now taken from Shark

Release 2.5-1

- Highlighting java/javascript syntax in fields for entering expression criteria (Transition conditions, deadline expressions, actual parameters)
- Activity's performer field now comes with higlight editor for java/javascript, and buttons to chose participant and variables
- Problems component view now depends on selection (Package/Workflow Process/ActivitySet)
- Improved expression's validation
- Implemented table sorting
- Improved D&D of table entries

Release 2.4-1

- Shark configuration adjustement to new shark 2.2-1version: PROCESS_ID and ACTIVITY_ID can be used in transition condition's and as actual parameters
- Implemented possibility to switch DataField to a FormalParameter and vice-versa
- Core extended to support new ext. attribute types per their parent object (Activity, Application, ...)
- Improved property panels
- Serbian property file renamed to have extension "sr" instead of "sh"
- Fixed bug with external packages (when newly opened XPDL has the external package with the same Id but different location as the one opened before)

Release 2.3-2

- Implementation-Version attribute included into MANIFEST of jar files
- Improved installation procedure - language dependent stuff and links to the application from start menu
- Fixed bug in SharkXPDLObjectFactory - wasn't able to properly use Activity's "Variables" tab
- Fixed: bug in ExternalPkgRelations component (NPE when selecting Arbitrary Expression Participipant)

Release 2.3-1

- Added new component for LDAP participant handling (the old action from the menu bar removed)
- Option buttons displayed for the language and configuration choices (the selected language/configuration is marked)
- Language switch is now possible in run-time
- Added configuration file which contains information about configuration name to be displayed for the configuration choices and in the title bar.
- Architectural changes to support add-on architecture
- Community version now can also display the result of References and Search action
- Core code is improved so now it is possible to add new object types without coding (just by configuration). Now we use XML templates for new object types (XPDL fragments).
- Top tree elements now have '-' displayed if they contain sub-elements
- Portuguese translation included
- Solved language usage issues with parser, JOptionPane, JFileChooser, missing strings,etc.
- Added additional language property file where you can define additional language specific properties per the configuration.
- Fixed: bug when adding external package which contains processes with activity sets and when "automatic layouting" is turned on.Fixed: bug in handling of transient packages that appeared when you have transient package in the pool and you try to open XPDL from unexisting file - this caused all packages to unload from XMLInterface and after it happens it was possible to have two XPDLs with same Id in TWE.

Release 2.2-1

- Implemented Search capability for searching XPDL elements based on specified criteria.
- In TWE's default configuration/mode, workflow_patterns.xpdl is present in Transient package pool
- Improvement: when configured to save XPDLs with namespace prefix and if XPDL didn't have xpdl namespace defined TWE now automatically adds this namespace.
- More icons for XPDL entities are shown in DetailedPackageNavigator view
- New configuration parameter for validating if variable (DataField or WorkflowProcess's FormalParameter) is not used anywhere in XPDL (nor as actual parameter or inside expression). If it is not used, warning is reported when validating XPDL.

- Icons defined for all entries in menu bar
- Some shortcut/accelerator keys have changed
- Expand all/Collapse all are now context sensitive in relation to selected tree-node
- Top tree elements now have '-' displayed if they contain sub-elements
- Switched to 2.9 version of xerces
- Spanish translation included
- XMLInterface changed - removed method getIdFromFile (it is now a static method of XMLUtil class)
- Code corrections based on FindBugs report
- Fixed: bug with reading of language property files from jar file - if there are spaces in the path where TWE is installed they were not read
- Fixed: bug in XMLCheckBoxPanel - whenever mouse was over, an apply button was enabled as if something has changed in a panel
- Fixed: bug that was introduced in previous version and appeares when there are external packages defined and Id of the main package is changed
- Fixed: when trying to save read-only document TWE didn't complain but the changes were not saved to a document
- Fixed bug in Shark configuration - bug fix in Shark configuration (related with deletion of ext. attribs)

Release 2.1-1-bugfix build 20061110

- fixed bug in Transient Package Pool componenet that was manifesting during usage of Common expression and Free Text expression participants
- fixed bug in Shark configuration - when you delete variable from 'Variable' tab, it was not removed from the table

Release 2.1-1

- added Transient Packag Pool component for displaying/adding/removing transient packages, and introduced configuration option to define transient packages that will be loaded by default on TWE startup
- added additional graph actions to select activities for selected transitions and vice versa
- improved shark configuration (added new features to better support shark professional version and new shark web admin application)
- improved documentation
- demo version gets demo documentation
- when creating new Package, xpdl namespace is added by default
- improved windows setup.exe distribution - user is asked if he wants to add desktop and qucik launch shortcut
- code re-organization

- Fixed: bug in XMLInterfaceForJDK13->getPackageById() method

Release 2.0-6

- property file location moved from %USER_HOME%/.JaWE to %TWE_HOME%/conf/xxx.
- enabled re-configuration at runtime - possible configurations are placed in sub-folders of %TWE_HOME%/conf, default configuration is written in %TWE_HOME%/conf/defaultconf
- beside default configuration, added shark configuration, wfmopen configuration, samples-loopactivity and purexpdl configuration
- shark configuration is adjustment of jawe to shark-engine restrictions + swing admin ext. attribs. Automatic change of ext. attribs when variable Id changes; validation on this special ext. attribs and validation according to shark restrictions
- wfmopen configuration is adjustment of jawe to wfmopen-engine restrictions + special ext. attribs. validation on this special ext. attribs and validation according to wfmopen restrictions
- improved simple graph layout action's algorithm (never goes back in the swimline whenever possible, when there is a split inside same swimline - new rows are created, handling of start/end bubbles improved)
- included David Delbecq's improved drop-down list (switching from menu-items to search-able list when there are 30 or more items) for inserting participants and selecting activity sets in graph component
- included David Delbecq's improved drop-down list (switching from menu-items to search-able list when there are 30 or more items) for selecting packages and workflow processes in info-bar
- Search panel renamed to References
- Documentation links in windows setup now reference full HTML document
- API entry in Windows dist is renamed to Together Workflow Editor Profesional API
- TWE stores new ext. attrib. information in XPDL for determining configuration used to create this XPDL
- added possibility to chose icon for the activity from the list of available icons
- added workflow_patterns.xpdl file (originally created for shark distribution)
- defined keyboard shortcuts for almost all menu items
- improved automatic insertion of start/end markers
- code re-organization (introduction of new classes and configuration properties)
- Fixed: rare bug in Graph component when start/end bubbles are the only one placed within FreeTextExpression participant

Release 2.0-5

- introduced possibility to chose when adding external packages if external package will be "transient" (it won't be saved into the file, but its purpose will be to provide a container where from you can copy common elements)
- introduction of compound icons (no need to define special icon for errors and warnings)

- Minor code changes/reorganization.
- improved automatical insertion of Start/End markers
- improved property panel for defining workflow relevant data: sub-panel for entering initial value is now bigger
- improved usage of LDAP connection: better support of ActiveDirectory search, added LDAPReferralHandling property, password field hidden
- Fixed: bugs in usage of LDAP connection for inserting participants (user+password authentication didn't work, search scope didn't work)
- Fixed: Graph component's tooltip bug in Zoom In/Out mode.

Release 2.0-4

- New possibility to save XPDLs with xpdl namespace prefix (TWE by default configured to do so)
- Added language property files for French and German language.
- Minor code changes/reorganization.
- Now using Xerces 2.8
- Fixed: problem with overlapping dialogs after loading XPDL file.
- Fixed: Enter key didn't work in a table panels

Release 2.0-3

- Added language property file for Serbian language.
- Code changes/reorganization.
- Fixed: if there were many participants in XPDL, pop-up menu for showing participants that can be inserted into graph couldn't show all of them.
- Fixed: TWE didn't change the references when the Id of TypeDeclaration is changed.

Release 2.0-2

- added WfXML authentication.
- default condition type for newly inserted transitions (via graph) is not set which means that by default unconditional transition is being inserted (previously default type was 'Condition')
- added new validation option called 'ValidateConditionByType' (which default value is true); when set to true, TWE will generate warning in following cases:
  - transition's condition type is not set (unconditional transition) and there is condition expression defined
  - transition's condition type is set to 'Condition' (conditional transition) and there is no condition expression defined
  - transition's condition type is set to 'Exception' (exceptional transition) and there is no condition expression defined

  - transition's condition type is set to 'Otherwise' and there is condition expression defined
  - transition's condition type is set to 'Default exception' and there is condition expression defined
- allowed deletion of invalid ExternalPackage elements (when there is no external package referenced)
- improved conversion of XPDL files from old JaWE versions -> Start/End markers are now properly positioned
- generally improved XPDL validation
- improved procedure of closing changed XPDLs - now TWE doesn't ask you to enter the file name if this is an XPDL opened from an existing file
- logging manager implementiation is now using Java logging instead system.out
- fixed NPE in LDAP connection
- fixed rare bug with repositioning of Objects in the table or list
- fixed bug with the displaying names of a certain extended attributes (e.g. if you defined ext. attrib. with name 'm', afterwards you'll see 'minute' instead of 'm' displayed in ext. attrib. property panel)
- fixed bug with Graph validation of Start/End markers during the save of Packages marked as 'Released' (TWE allowed saving of incorrect package)
- fixed bug with XPDL validation of Tool's Actual->Formal parameter mapping during the save of Packages marked as 'Released' (TWE allowed saving of incorrect package)
- fixed problem with de-selection of graph during undo

Appendix A. GNU Free Documentation License

GNU Free Documentation License
Version 1.3, 3 November 2008

Copyright (C) 2000, 2001, 2002, 2007, 2008 Free Software Foundation, Inc.
<http://fsf.org/>
Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

The "publisher" means any person or entity that distributes copies of the Document to the public.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
D. Preserve all the copyright notices of the Document.
E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
H. Include an unaltered copy of this License.
I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If

there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.
N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.
O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties--for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING

"Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.

"CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

"Incorporate" means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is "eligible for relicensing" if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

```
    Copyright (c)  YEAR  YOUR NAME.
    Permission is granted to copy, distribute and/or modify this document
    under the terms of the GNU Free Documentation License, Version 1.3
    or any later version published by the Free Software Foundation;
    with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.
    A copy of the license is included in the section entitled "GNU
    Free Documentation License".
```

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with...Texts." line with this:

```
    with the Invariant Sections being LIST THEIR TITLES, with the
    Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.
```

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.