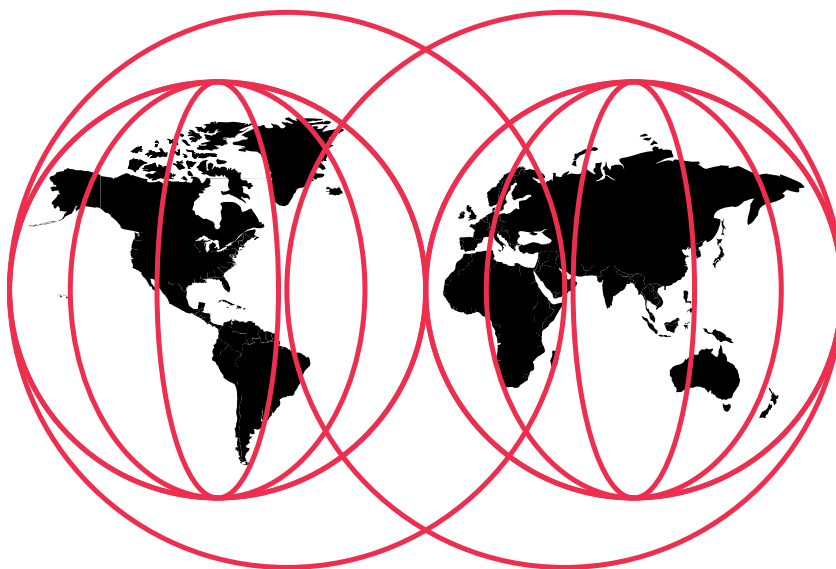


Using Domino Workflow

Søren Peter Nielsen, Carol Easthope, Pieter Gosselink, Karl Gutsze, Janno Roele



International Technical Support Organization

www.redbooks.ibm.com



International Technical Support Organization

Using Domino Workflow

May 2000

Take Note!

Before using this information and the product it supports, be sure to read the general information in the Special Notices section at the back of this book.

First Edition (May 2000)

This edition applies to Lotus Domino Workflow 2.0.

Comments may be addressed to: IBM Corporation, International Technical Support Organization
Dept. HYJ Mail Station P099
2455 South Road
Poughkeepsie, NY 12601-5400

When you send information to IBM, you grant IBM a non-exclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

© **International Business Machines Corporation 2000. All rights reserved.**

Note to U.S. Government Users: Documentation related to restricted rights. Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corp.

Contents

Preface	ix	2 Domino Workflow architecture ..	15
The team that wrote this redbook	ix	Basic concepts of Domino Workflow	15
Comments welcome	xi	Binder concept	16
1 Workflow	1	Activities and their owners	16
Workflow defined	1	Routing relations: defining the flow	18
Main workflow components	2	Process map: visual process definition	19
Summary of main workflow components	3	Potential activity owners versus activity owners	20
Getting structure into your work	4	Using groups and roles	21
The life cycle of workflow	6	Processes and their instances: jobs	21
Why use a workflow tool?	7	Managing existing processes: job owners ..	22
Workflow support in Lotus		Team concept	24
Domino/Notes	7	Parallel paths: splitting up your binder ..	24
Transparency of the process	8	Basic architecture of Domino Workflow	25
Visualizing your business processes	8	Working environment: the workflow engine	25
Managing your processes	9	Designing processes	31
Change management	9	Running your jobs	34
Separating workflow development and application development	10	Enhancements to the basic architecture	37
Where should I use a workflow tool?	10	Connecting all the parts	37
What is Domino Workflow?	11	Using multiple applications	39
Business process orientation	11	Using alternative organization models	41
Activity orientation	12	Summary	42
People orientation	12	3 Installation and basic setup of Domino Workflow	43
Document orientation	12	Overview of the setup procedure	43
Integration into Lotus Domino/Notes	13	Before you begin	44
Integration with other systems	13	Domino server requirements	44
Summary	14		

Installing product components on the workstation	45	The first prototype: a basic scenario to handle customer requests	62
Choosing the components to install	46	Defining the tasks to process a customer request	63
Creating your work environment on a server	47	Identifying the workflow participants in the basic scenario	63
Creating Domino Workflow databases on the server	47	Specifying when the basic application tasks are performed	64
General database creation procedure	48	Implementing the “Who”: define your organizational units in the organization directory	65
Enabling and signing Domino Workflow Engine agents	49	Implementing the “What”: design your forms in the application database	71
Enabling the agents in the Application database	49	First prototype continued: implementing the “When” using the Architect	74
Enabling the agents in the Organization database	49	Creating a database profile and opening the databases	75
Configuring the Organization Directory database	50	Creating a new process	77
Creating the setup document	50	Setting the basic process properties	77
Configuring the setup document	51	Creating and defining the process activities	81
Configuring the Application database	53	Defining the create request activity	82
Creating the setup document	54	Defining the research request activity	86
Configuring the setup document	54	Defining the create response activity	88
Setting the Access Control Lists for the databases	55	Saving and activating the process	90
Quick start ACL settings	55	Second prototype: adding more basics	91
Populating the Organization Directory with person data	56	Prototyping strategies	91
Importing people or groups from Domino Directory	56	Creating forms and process copies for the second prototype	93
Copy and paste from Domino Directory	57	Making changes to the process and activity properties	95
Review your basic setup	59	Third prototype: getting more difficult	103
Summary	59	Getting started with the prototype	104
4 Defining your first process	61	Modifications to the create request activity	105
What is involved in process design	61	Changing the routing relations	105
Preparing for the process design: the three Ws	61	Enabling other new documents in job binders	107
What is involved in implementing your solution	62		

Modifying the create response activity . . .	109	6 Introducing Domino Workflow	
Finishing up the third prototype	110	in an existing application	141
Fourth prototype: nearly there	110	General steps to add Domino Workflow to	
Defining the marketing research		your application	141
approval cycle	111	Overview of the process in the existing	
Defining the response approval cycle . . .	115	application	141
Defining the reply to customer		Creating the review process	142
activity	119	Review process definition	142
Finishing up	119	Organization database	143
Fifth prototype: last version	120	Customizing the Document Library	
Defining the automated send		template	147
acknowledgment activity	121	Summary	150
Adjusting routing and binder joins	126	7 A closer look at Domino	
Finishing up	127	Workflow: process design and	
Summary	127	implementation	151
5 Accessing Domino Workflow		Using existing elements: Business Object	
from Web clients	129	Library	151
How to make your workflow database		Custom attributes	154
available to Web clients	129	Standardizing corporate routines:	
Use of the Domino HTTP		subprocesses	155
server	130	Using a subprocess	156
Enable your application forms	130	Customizing Domino Workflow using the	
Activate the Web agents	130	Developer's Toolkit	157
Adjust the setup document	132	Various ways to initiate your job	158
Update the process cache	133	Standard initiation of a new job	159
Accessing the workflow database through		Advanced initiation	160
the standard Web interface	134	A practical initiation scenario	165
Advantages and disadvantages of the		Realization of the scenario	166
standard Web user interface	137	Naming your job automatically	168
Advantages	137	By a standard initiation	168
Disadvantages	138	Using workflow events	168
Conclusion	138	Basics of automating an activity	169
Customization of the Domino Workflow		Sending a mail message	170
Web user interface	139	Executing a server program	170
Summary	140	Starting a LotusScript agent	171

Additional scenarios using an automated activity	174
Automatic claim of activities	175
Supporting a family: parent-child processes	176
Summary	178

8 Using the organization directory 179

Why use a separate organization directory	179
Conclusion	182
Advanced use of the Domino Workflow organization directory	182
Relations	183
Job properties	186
Resources	188
Alternative organization directories	191
Integrating an alternative organization directory	192
Mapping files	192
Adding your own OrgaTypes to the Architect	194
Replace design elements based on the Toolkit	195
Change your Setup document	196
Change your data sources in the Architect	197
Change and reactivate your processes	197
Integrate with other organization directories	198
Summary	199

9 Distributed processing 201

Multiple databases	201
Process subsets	201
Subprocesses	202
Multiple replicas	202
Server cluster	203
Local replicas - mobile users	203
Using an agent server	205

Location replicas	205
Selective replication	206
Handling replication conflicts	207
Summary	208

10 Integration 209

Domino.Doc	209
Access profiles	210
Initiating workflows	211
Archiving	211
Automated activity search	212
Automatic check in/check out	213
Linking between Domino.Doc and Domino Workflow documents	214
Restrictions	214
XML	215
XML defined	215
Domino Workflow and XML	215
Configuring Domino Workflow	216
Summary	217

11 Running Domino Workflow applications 219

Make changes at run time	219
Change the team working on an activity	219
Introduce some new activities (reassign)	223
Change the route of your job (reroute)	225
Cancel a job	227
Keep an eye on the jobs	228
The most important views on workflow	229
Get all details of one specific job (process Viewer)	233
Recording your job context (audit trail)	235
Using events to customize the audit trail	236
Managing the audit information	236

A completed job (archive)	237
Setting up an archive	237
Various ways of archiving	239
Optimizing your workflow application	241
Design elements	242
Performance issues	243
Troubleshooting	247
Some common errors	247
General tips	250
Summary	252
12 Domino Workflow and LotusScript	253
Automated activities	253
Automation agents	253
Redbook Binder classes	255
Binder	256
Binder document	256
Extending the classes	256
Redbook automation agent class	257
Redbook automation agent	257
Examples of automation agents	258
Unsuccessful automation agents	259
Events	261
Changes to the event library	261
Summary	264
Appendix A Parent-child processing	265
Scenario	265
Concepts and features used	265
Steps to make it work	266
Sample code for Parent Spawn agent	267
Sample code for Child Post Back Response	270

Appendix B Advanced Web use of Domino Workflow	275
Before starting to build your Web user interface	275
Programming the Web solution	276
Views in the application	279
The “Desktop” views	282
Change the standard views	283
How to add or remove views	284
The Workflow Navigator	285
Starting new jobs	285
Binder documents / Domino Workflow Information Subform	286
Other functionality	289
Adding your own code for activity validations	289
Adding custom Actions (Web style subform only)	290
Summary	290
Appendix C LotusScript sample classes	291
Redbook binder library	292
RedbookBinder Class	292
RedbookBinderDocument Class	295
Redbook automation agent library	296
RedbookAutomationAgent	296
Deactivated automation jobs view	296
Redbook automation template agent	297
Redbook activate waiting jobs agent	297
Special notices	299
Additional Web material	303
How to get the Web material	303
Related publications	305
IBM Redbooks	305
Other Lotus-related IBM Redbooks	306

IBM Redbooks collections on CD-ROM	308
Other resources	308
Web sites	309
How to get IBM Redbooks	311
IBM intranet for employees	311
Index	315
IBM Redbooks review	319

Preface

In this book we describe how to support your business processes with Domino Workflow™. We do this with an example scenario, to which we add enhancements and variations throughout the book. First, we describe, step by step, a full implementation of a simple process using the built-in Domino Workflow functionality. Next, we show how to enable our application for Web access, before walking through a case study where we enable an existing Domino application with Domino Workflow. In the latter part of the book we discuss topics like customization, advanced techniques, the organization directory, and integration with other systems.

The examples used in this book are available on the IBM Redbooks Web site.

This redbook is written for Domino Workflow solution designers and programmers, customers, Business Partners, and other members of the IBM and Lotus community who need a good technical understanding of how to use Domino Workflow.

The team that wrote this redbook

This redbook was produced by a team of specialists from around the world working at the International Technical Support Organization Center at Lotus in Cambridge, Massachusetts, USA.

Søren Peter Nielsen works for the International Technical Support Organization at Lotus Development, Cambridge, Massachusetts. He manages projects that produce redbooks about all areas of Lotus products. Before joining the ITSO in 1998, Søren worked as an IT Architect for IBM Global Services in Denmark, designing solutions for a wide range of industries. Søren is a Certified Lotus Professional at the Principal level in Application Development and System Administration.

Carol Easthope is a Principal Systems Engineer with the Architected Solutions Group, Lotus Professional Services, Cambridge. In the five plus years Carol has been with ASG, she has participated in the design and development of large-scale Notes/Domino/Web applications for clients in the financial, energy, manufacturing, and computer technology industries. Carol joined Lotus in 1991, as a research engineer engaged in the research and prototyping of workgroup and collaboration products.

Pieter Gosselink is an IT Specialist at AT&T Global Network Services Netherlands. Pieter has over eight years of experience in application design and development, including three years with Lotus Notes and Domino. He graduated in Computer Science at Twente University, Netherlands.

Karl Gutsze is a consultant with Mettenmeier Business Services (MBS), a German company specializing in development and implementation of workflow-based solutions. Before this he spent three years with ONEstone (a company that developed the prototype of Domino Workflow), first as part of a quality assurance team and later on leading the market research and technical evaluation of workflow products. He holds the master's degree in Wirtschaftsinformatik (Business & Computer Science) in Paderborn, and is also a CLP in Application Development.

Janno Roele is a technical Domino Workflow specialist working for Workgroup Enabling Business Solutions at Origin Netherlands. He joined Origin after earning the master of science degree in Information Management at Tilburg University. Janno has over two years of technical and functional Domino Workflow experience. He did several projects developing and implementing Domino Workflow applications for industries like insurance, manufacturing and petroleum. Besides this Janno is also a Certified Lotus Professional in Application Development.

A number of people have provided support and guidance. We would like to thank the following people from Lotus (unless otherwise noted):

- Markus Dierker
- Stephan Gieffers
- Niklas Heidloff
- Ralf Heindoerfer
- Wolfgang W. Hilpert
- Judy Jalbert
- Claudia Sauer
- Lisa Toussaint
- Cees van der Woude
- David Morrison, ITSO Cambridge
- Alison Chandler, ITSO Poughkeepsie
- Graphic Services, Lotus Cambridge

Comments welcome

Your comments are important to us!

We want our redbooks to be as helpful as possible. Please send us your comments about this or other redbooks in one of the following ways:

- Fax the evaluation form found at the back of this book to the fax number shown on the form.
- Use the online evaluation form found at <http://www.redbooks.ibm.com/>
- Send your comments in an Internet note to redbook@us.ibm.com

Chapter 1

Workflow

This redbook describes the Domino Workflow product. Before we get into the specifics of this product, we need to explain some general terms related to this subject as we understand them. The most important of them is the term *workflow*, which has recently come into common usage, and which is used — and misused — in various contexts. Sometimes the only purpose of speaking about workflow is strategic; sometimes it is merely a buzzword. As we consider it to be much more than that, we want to introduce workflow as completely as we can. This chapter includes the following topics:

- The most important characteristics of workflow
- The advantages of using a standardized workflow product
- How Domino Workflow fits into the general workflow context

Workflow defined

There are many ways to define workflow. There are also many applications and solutions which offer workflow and which attempt to define workflow based on their own feature sets and value propositions. None of them is totally right or totally wrong. Nevertheless, many common elements of each definition can be easily recognized. Since this redbook, like all redbooks, doesn't try to cover the subject on an academic level, we provide a pragmatic working definition that will allow you to understand the subject as it is and that doesn't pretend to be the best and only one. It should help you to understand some important aspects that have to be considered before using workflow. Specifically, in this section we discuss:

- The main workflow components:
 - Business processes
 - Information
 - People
- Different types of structured workflow processes
- The life cycle of workflow processes

Main workflow components

Before you consider workflow, you have to give some thought to the organization of your company. You can define what your company does through its goals. Some action has to be taken in order to reach those goals. Such action usually involves fulfilling a number of activities or tasks and requires cooperation of multiple organization members. That is what workflow is about: it concerns the coordination of people, information objects, events that cause work to flow (for instance, deadlines) and the state or status of work required to meet business objectives.

Business processes

Business processes describe what things have to be done, how they must be done, and who should do them. They are like recipes in a cookbook that define how business objectives can be accomplished. This approach has been very well researched in the production factory environment, but was largely ignored in the office world, partly due to missing technology. In the last ten years technologies have emerged to support this kind of approach in the office environment. Today, office workers spend the main part of their time using computers plugged into networks with access to LAN devices and the Web. Communication between various systems is possible, as well as the ability to automate and facilitate work are there. Timely communication between people and information objects, based on business rules, is what workflow is all about in the office environment today. If you organize your work in this manner, you are essentially already dealing with workflow.

To successfully automate the flow of work in your company, you must select the appropriate business processes on which to apply structured workflow. Such a selection involves evaluating several other factors about the company: its goals and priorities, and the nature of the information associated with the particular business process to be automated.

First, consider the goals and priorities of your company. You should concentrate on automating processes that matter, that make a big difference to your company. Your goals should be clearly set and prioritized. It is a good practice to include some measurable values, like speeding up your process from two weeks to eight days or increasing the throughput of one person from two claims per hour to three. Every workflow has an entrance and an exit point; and an overall goal that should be reached between those two. It also describes the way in which things have to be done. It can't give you a ready recipe for how you should do things, but it is a helpful tool for the realization of your ideas.

Information

The second consideration, when selecting which of your business processes to automate, is the information associated with the process. This is a more technical issue, and it relates to your infrastructure and environment. You should choose processes where some clear patterns are visible and you can precisely describe them. You should also be able to clearly define the context in which the work can be done. Such an approach allows for the quick automation of some steps. Workflow can exist only in cases in which you can at least partly define the structure of the information that is required to finish your job properly.

People

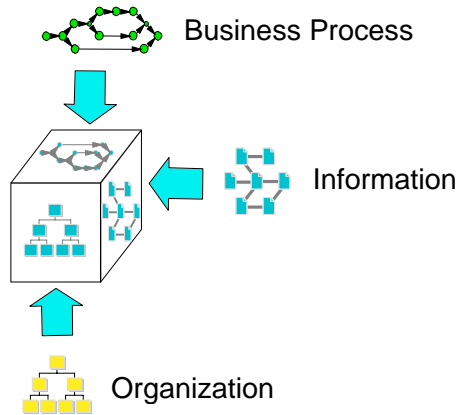
The most important component in most workflows is people. They are the ones who create content, make decisions, delegate work, and oversee the completion of work. It is not the purpose of workflow automation to make the users' efforts more complicated, but rather to help them concentrate on the important parts of the business process. People shouldn't have to work at maintaining workflow; workflow should evolve around them. This approach differs significantly from the industrial one, where the production processes are very stable and don't require human interaction in the realization phase. In contrast, in an office environment workflow has to be built around human actors assuming various roles in various situations. Therefore, it requires well-defined roles and organization relationships.

Summary of main workflow components

Combining all of the factors described previously yields a general definition of workflow as:

- Partly automated business processes that achieve a given goal in a predefined way
- Automatically presenting required information to the right group of people at the right time
- Automating strictly repetitive parts of the work

This definition is also illustrated in the following figure:



There is much more to workflow than the points covered by this definition. Among the other significant issues are:

- Controlling existing processes and optimizing them
- Interdependencies between processes and enterprise-wide infrastructure standards
- Concerns about the fact that business rules and responsibilities need to dynamically change with the business environment

Each of these issues, as well as other considerations, will be discussed in detail in later sections. The differences between a standard workflow tool and a custom-made workflow are also explained.

Finally, it is quite impossible to draw a clear line between workflow and other enterprise resource and knowledge management issues. They are all about your company, and sometimes cover the same things from various perspectives. However, the workflow perspective will help your company to make the next step forward successful.

Getting structure into your work

There are various scenarios concerning workflow. Some of them require more structure, some less. Most processes can be assigned a place in a workflow continuum that tries to classify processes according to their structure. Knowing what type of process you are dealing with helps you to pick the right workflow solution for it. In this section we describe the features of four process types:

- Structured processes
- Semi-structured processes

- Cooperative processes
- Real-world processes

Structured processes

Structured processes are the best-known processes. You can find them in any production plant, where it is very well defined which machine has to work on which part of the process, the production times are set precisely and the production workers are assigned well-defined jobs. Such processes can also be found in some offices that deal with mainly repetitive jobs that have to be controlled and documented according to strict rules. Examples of this include the high volume claim processing done by an insurance company and license-issuing activities performed by governmental departments.

In some parts of such well-structured processes some exceptions can be allowed. It helps to make them flexible. Nevertheless, all exceptions have to be controlled and documented. Examples of such exceptions include a manager having the ability to override someone's decision, or an exceptional consultation request being made at one of the process steps. Such exceptions don't change the process, and they are not recurring. If they are, they should be explicitly included in the process.

Semi-structured processes

In many cases only the overall process structure can be defined in advance, but certain activities may require that people participate in the process in a manner in which every step of their work is not predefined. For instance, a graphic design group creating an identity for a new product may work in a completely collaborative manner, and their approach and team members may vary with each project they are engaged in. The new product launch process also has many activities that are based on predefined business rules and assigned to persons based on a role or position within the organization. For instance, the legal search activity to determine whether the new product name is copyright and trademark free is a structured step in the process. The ability to support both structured and collaborative work in a process is important for many strategic business processes.

Cooperative processes

This group of processes has the least predefined structure of all described types. You make most of the decisions at the time of execution, and they are based on your personal experience and knowledge of your organization. Such processes can be based on e-mail or some other kind of a shared environment, such as database or Web space. Such processes don't pose any restriction on the creativity of the participants, but are very difficult to handle or to control. They can usually be found in small companies, where processes evolve around the individuals participating in them. Such processes usually cause

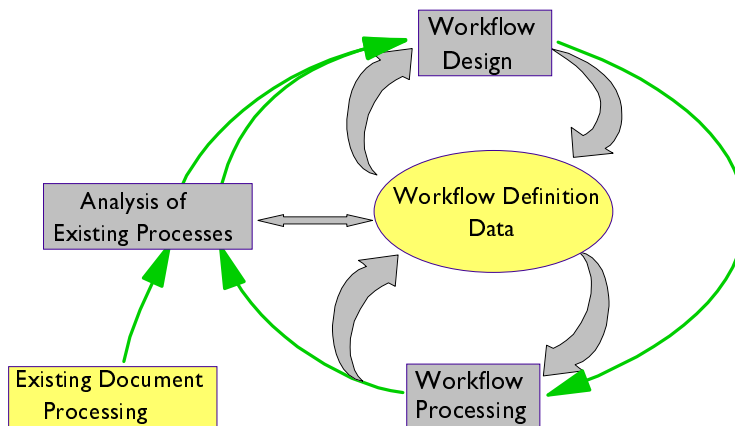
very big problems after organizational changes, such as when someone leaves the company or joins it. Therefore, many companies try to migrate the spontaneous processes to a more structured form.

Real-world processes: mixed forms

All those classifications are very theoretical, and for the most part not exactly applicable to real business processes. Nevertheless, you can probably at least partly assign your processes to one such category. You should also be able to assign a category to various groups of tasks in your processes. It is good practice to do this, both to better understand your requirements on the workflow, and to choose which parts will be easy to automate and which have to be defined first. Keep in mind that workflow will never be better than your business practices. If you have problems within your company, workflow may help you to visualize them and to implement some changes, but you'll still have to decide what those changes should be.

The life cycle of workflow

Every company is a living organization and therefore it undergoes constant changes. If it doesn't change, it dies. It has to grow, to mature, to learn, and, at some point, to find some ways of managing the constant changes in order to react to a changing environment. All the business processes have to accompany these changes. The reason for it can be just a simple change in the organization, a rationalizing action, or a constant improvement process. So, because business processes change over time, your workflow also has to change.



As already mentioned, you don't start running your company by creating workflow. You run your company by getting things done. By doing this you create some documents, and you invent ways of getting them processed. Therefore, you do have some existing document processing.

To create workflow, start by analyzing your existing document processing. This analysis will lead you to some kind of definition of your processes. This is where the workflow starts. From the definition of the processes you can identify recurring patterns that are your first workflow design. You create workflow definition data by storing this data in any repository. The next step is automating some parts of your processes and automating the transition between the process steps. After you have done this, you have a functional workflow that has to be evaluated in practice.

The first part of the evaluation is always theoretical: you can introduce some changes in the existing processes to make them more efficient. The second part of the evaluation takes place after you have worked with the automated processes (the workflow) for some time. In both cases you return to analysis of your workflow in order to make it better. After this the cycle starts from the beginning. It is a never-ending cycle of changes that makes your company better all the time. Concentrating on workflow makes the changes a natural thing. It helps you to redefine your goals and to make your processes more suitable for reaching your company's goals. It also helps the organization to grow and mature.

Why use a workflow tool?

Once you have decided to automate some parts of your processes, you have to choose a strategy for doing it. The two main choices are doing it all yourself or using a standardized workflow tool. In this section we compare the two approaches when building on top of Lotus Domino/Notes and show some of their good and bad points. The criteria we consider for each approach are:

- Transparency of the process
- Visualization of the business process
- Management of the process
- Change management for process change
- Keeping process definition and application logic separate

In addition, we give some advice about how to identify processes where a workflow tool can help.

Workflow support in Lotus Domino/Notes

Lotus Domino/Notes already offers a lot of simple mechanisms that support workflow. You can create workflow fields and rules in a Domino application using the Domino Designer, and for simple, unstructured processes this may be the most cost-effective approach. However, building applications that support more complex business processes using this approach comes at a cost.

It is very easy to define a review cycle on a couple of documents, but if you want to transfer your deployment to some other application, you have to do the work over again. If the logic of your review cycle changes, you have to go into various documents and change field values or LotusScript code. If you want to keep an eye on your processing cycle you have to know the right field values. All this causes no problems as long as you are dealing with simple processes that don't change a lot. However, if you try to manage complicated activities, the complexity of your application grows exponentially and your application quickly becomes difficult to maintain. This is the point where you should consider using a workflow product. The real question is not *if* you can automate your processes with Lotus Domino/Notes, but *how much effort* would it take to create and maintain the workflow using this method.

Transparency of the process

As mentioned previously, workflow is not a goal in itself; its purpose is to help you manage your business processes. Therefore, you actually don't need to know how things are done from a technical point of view. You want to concentrate on your work, on the application you are working with, and on the documents and their content. How your workflow is managed technically is something that a workflow tool can hide from you. Such a tool lets you allocate your skills to the development of the important part of your processes — the part that is unique for your company and can't be found somewhere else. A standard tool lets you interact with workflow mechanisms through well-defined interfaces, and hides the internal fields and code that gets the work done for you. It's the same approach Lotus has taken in providing you with customizable, ready-to-use templates. Skilled customization is a challenging task, though, and no Domino/Notes developer would try to do everything on their own. The workflow field is much more complex; that's why you have to buy a whole product instead of using a simple template. A workflow product lets you concentrate on business and workflow instead of bothering with internal fields and formulas.

Visualizing your business processes

One of the really big advantages of a process-oriented company is its ability to see the content of the work in its context. The most important context of work is its integration with overall goals. The best way to define this is to set the goals, describe the processes leading to them, and then describe the activities included in those processes. You can visualize it using any kind of chart that is appropriate to your activities. Regretfully, there is always a big gap between such charts and the way things actually get done. When creating a chart you can't foresee the technical problems you may encounter implementing it. There can also be a big gap between the time you create a chart and when you implement the defined process. This is another reason for using a standard workflow tool. You can visualize your processes and be sure that they can be

deployed the way you have drawn them. There will always be a small time period between drawing your chart and implementing a process, but it will be reduced to the minimum. You only have to concentrate on the forms you need and their logic; you do not have to worry about the workflow logic as well. You will be able to present the processes as they are to their users, reducing the introduction phase.

Managing your processes

After you have developed your workflow, you have to maintain it. You want to be able to see who is doing what, at which stage a specific process instance is, or what has to be done in the next step. In order to reach such a goal you'll have to create some views and agents. It takes a lot of time to specify your needs and mechanisms to fulfill them. A standard workflow product comes with such views and the mechanisms right out of the box. You will probably want to customize some of them or add new ones, but you can use the interfaces provided for you. It won't be necessary to keep a long list of elements you created, as you can find such a list in the documentation. You will also be able to delegate the maintenance to someone else, documenting only the changes you have made. You will also gain experience that is useful the next time you need to use the tool. The basic knowledge is easy to transfer and the specific knowledge can be quickly enhanced and leveraged.

Change management

As the processes go through their life cycle, they change. In production environments such changes are pretty rare, because they are restricted by technology and machines. In the office environment, where people are doing their job using flexible tools, such changes can happen frequently. Therefore, good support for all kinds of changes has to be integrated into every workflow solution. If you are planning to do your own workflow solution, you have to document all fields and actions that store or use relevant information, and you have to be aware of all their interdependencies. Even minor process changes can force you to rewrite a large section of your code. You also have to develop some strategy for transferring existing processes into a new version. It's a lot of work and you create a lot of overhead, but it has to be done. Once more, an existing workflow product can save time and effort. Even if you customize it, the basic architecture will be taken care of, and in the first deployment stage the basic mechanisms of the tool might even be adequate.

Separating workflow development and application development

We have already mentioned the importance of separating business processes from workflow implementation. This is such an important concept that it bears reinforcing here. Lotus Domino/Notes allows you access to your code at any place. It has some advantages for implementing small local changes, but it can make documenting an application very difficult. In a process-oriented company you'll want to have a clean environment, in which it is easy to tell where the code is and what it does. It is the best way to efficiently manage your processes and their changes. Because of this you will have to put your workflow-related code and design elements in some well-defined places to separate them from the rest of your application. By doing this you will end up essentially developing a workflow product. Why should you do something that has already been done? It is much easier to deploy an existing product. If you are already using some templates and script libraries, the next logical step would be to use a workflow product. Then you can easily concentrate on the application development and use the time you intended to invest in developing the workflow part learning the handling of the workflow product instead.

Where should I use a workflow tool?

While this question looks like a simple one, it takes some consideration to answer it correctly. A workflow solution can be used to automate any business process, but it might not always be the best choice. Consider the answers to the questions listed below to better define your requirements:

- Can you identify some recurring patterns in your process?
- Do you pass your documents between various groups of people?
- Can you identify some context for the work?
- How often do your processes change?
- How long does it take to accomplish one instance of your process?
- Can you identify some recurring simple tasks?

Workflow is a very powerful paradigm to support dynamic processes containing reoccurring patterns. Therefore, it is crucial to identify them. If you can't identify any structure, it doesn't make sense to use workflow tools.

Workflow can automate passing the documents between various people or groups of people. If all the work is done by one person, it will only put some restrictions on this person without being of much help.

Workflow lets you present some context to a specific person dealing with specific work. If you can specify this context, you'll be able to present it at the right time, speeding up the work.

Though workflow supports constant change, you will encounter problems if dealing with processes whose structure changes more quickly than the processes can be finished. If, for example, your process of building an apartment house lasts three years from planning to selling the apartments, and the legal restrictions change every year, you will end up having a lot of problems. You will have to migrate your process to a newer version several times. In such a case it is preferable to create smaller processes containing tasks like making first drafts, obtaining building permits, getting some investors, actual building, and selling the apartments. You can chain your processes into a big one or put them in some kind of a container from which they will be triggered; this way your processes will run according to the actual restrictions. The recurring simple tasks are the perfect candidates for full automation.

There are also workflows that concentrate on the full process automation. In such a case it is wise to use a workflow product that focuses on the automated flow of work between various machines and applications and is not people-oriented (like Domino Workflow is), such as MQSeries Workflow. Workflow products can simply use a different workflow definition and therefore aim at different goals. Getting the best of each kind of product and combining them can lead to a very high level of efficiency without restricting people from doing what they can do best: deal with highly complex situations using their experience, judgment and creativity.

What is Domino Workflow?

In the first part of this chapter we introduced some characteristics of a workflow system, in the second part we discussed the advantages of a standard workflow tool, and in this section we focus on some of the main concepts underlying the workflow tool that is the subject of this book, Domino Workflow. A conceptual overview of the product is included here; the architecture is discussed in the next chapter, and the technical issues are presented in the rest of this book.

Business process orientation

The main focus of Domino Workflow is business processes. Therefore, if you want to introduce workflow into your organization, you should start by focusing on the ways in which work gets done in your company. Don't be concerned with the technical implementation details at first; instead concentrate on your company, your business processes, and the way to get things done more quickly and efficiently. Domino Workflow provides an organized approach to help you think about your company's goals and the best way to achieve them. To help you visualize what are otherwise abstract concepts, the Domino Workflow Architect lets you draw your processes as a flowchart of activities that have one starting point and

proceed towards a specific goal. Drawing such a flowchart lets you not only define your workflow, but also helps you to focus on the important things—your business processes.

Activity orientation

To complete a process various things have to happen. Such things are represented in Domino Workflow as *activities*. Activities are groups of tasks that lead towards reaching an overall goal of the process. Each such group of tasks is performed by one person and can include additional sub-goals that should be reached by this person. All activities are connected through logical relationships that define the order in which they have to be worked on. As some parts of the work can be done automatically, there are also automated activities in this model. Such an approach lets you easily understand what happens in your business process, why it happens, and what the previous and following actions are. Passing the right information between activities and putting it into the right context are key capabilities of Domino Workflow.

People orientation

As we mentioned previously, introducing workflow in an office environment is different from automating a production plant. You don't want to automate the way people think, and you don't want to deprive them of their creativity. Domino Workflow has been designed with a people-oriented approach to workflow: you have to put people working on problems in the middle of your picture. Domino Workflow doesn't interfere with the way you do your job — it takes away the burden of deciding what to do after you have finished your part of the work. It may also help you to automate some parts of the process, but that's not its main focus. Domino Workflow is about improving organizational effectiveness and knowledge management and not about, for instance, automating the way your fax gateway works. You can try to use it for such an automation also, but you shouldn't focus entirely on that. Domino Workflow is at its best bringing structure into areas in which Domino excels: organizing the work of various people, helping them to cooperatively work on shared goals, and coordinating their efforts to reach their goal. Using it for such a purpose lets you leverage its possibilities and guarantees you a lot of satisfaction by reaching your goals. If you can put people in the middle of your processes, you can use Domino Workflow.

Document orientation

As Domino Workflow is a product that deals with the business processes taking place in the office environment, it is document-oriented. It inherited this approach from Domino/Notes. Documents are very natural things for people to work with, as they have been dealing with them all their lives. What's more, documents let you put all the relevant changing information

into the context of some more static information concerning the task you want to perform at the given moment. Domino Workflow moves the documents around, passing them to the appropriate people, taking care of the access rights and letting the work move smoothly. Work tends to have more complex content than just a single document: workflow allows you to do all the operations on a whole set of documents connected logically into binders. The way Domino Workflow does all these things is described visually at the process modeling stage, so you don't have to bother with these actions on the Domino/Notes level. All this logic is hidden from the users working in their normal environment — except for the elements you want them to see, and mechanisms allowing them to influence the course of actions defined.

Integration into Lotus Domino/Notes

One of the most important strengths of Domino Workflow is its integration with Domino/Notes. Domino Workflow is fully integrated into this platform, leveraging all its capabilities. The parts used to define processes and their logic are not essential to deploying the product at run time. At run time you are dealing only with pure Domino/Notes databases containing some design elements, scripts, and agents. That's why you don't have to worry about integration issues. If you need to connect to any other product to which you have successfully connected Domino, it will work exactly the same way with Domino Workflow. You don't need to worry about running any additional tasks on your server: Domino Workflow uses only some LotusScript agents. The forms you use are Domino/Notes forms, the actions and scripts as well. You can use the knowledge about Domino/Notes you already have, so it won't be necessary to learn any new programming techniques. You will have to understand the architecture of Domino Workflow to make the most of it, but as soon as you have done so, you'll be able to build many new features yourself. You will always stay in the same environment. For this reason you can build workflow for Domino/Notes on your own. Thus, Domino Workflow is an enhancement to the Domino platform using only the features of the platform. On the other hand, you shouldn't forget how much time and effort it takes to develop such a complex tool.

Integration with other systems

As we already mentioned, Domino Workflow can be integrated with any other system that can be integrated with Domino. It does not need its own interfaces, but uses those of Domino. There are real-world examples of integration with SAP R/3, Domino.Doc, Domino Enterprise Connection Services (DECS) and other systems. A later chapter in this book is devoted to the topic of integration, mainly to show that it is nothing to fear.

Summary

In this chapter we presented a discussion about workflow, the kinds of problems it deals with and some of its most important aspects. We described the workflow approach taken by Domino Workflow. We also explained why it may be wise to use a workflow development and management tool instead of programming it yourself. Finally, we introduced Domino Workflow to you in a very general way.

Chapter 2

Domino Workflow architecture

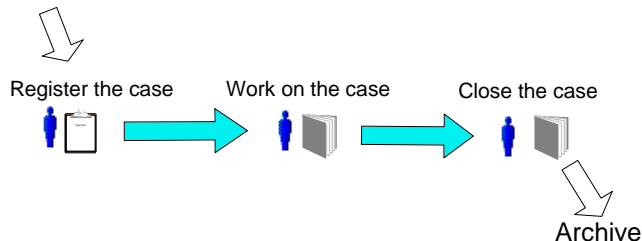
In this chapter we discuss the architecture of Domino Workflow. To do this we will introduce some basic concepts used to create workflow, the components required to deploy workflow along with their interdependencies, and some scenarios that you might use in order to customize the architecture according to your needs. We don't intend to describe all the concepts and dependencies, as you can find them in the product documentation. Much more important for us is to establish a general understanding of Domino Workflow and the way that it works.

Basic concepts of Domino Workflow

We start our discussion by introducing a simple, general scenario that shows how the work gets done. This scenario does not show any real situation. It is purposely abstract, to best introduce the terms of Domino Workflow and its basic architecture. We can introduce this concept more easily only in the context of an office environment. Here is the scenario:

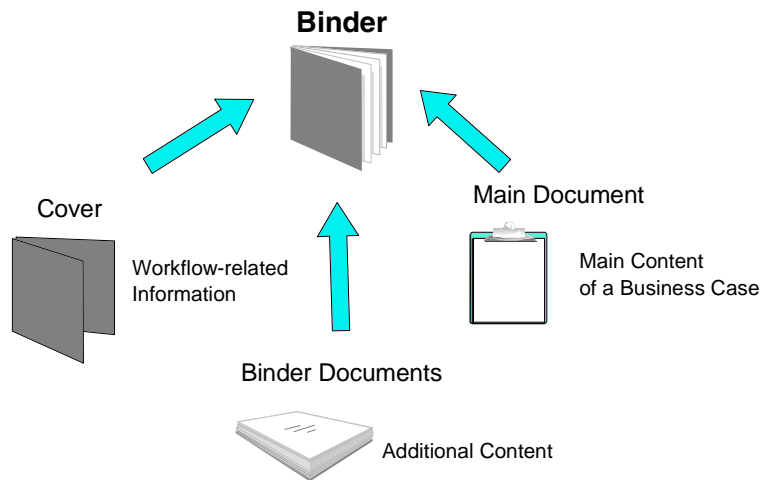
Something happens. You create a document to write down what happened. If it's required, you pass this document to someone else. The next person works on the document, and afterwards passes it on to someone who will work on it further. Some time later, after the work is done, the case is closed and the document gets archived. It is a typical way in which things get done in the example company. We can illustrate it with a simple figure:

Requirement



Binder concept

It may also happen that the content of the work requires more than one document. In that case you can create a binder consisting of multiple documents that will be passed between people. This is also how Domino Workflow works. You have binders consisting of a logical collection of documents being passed between people. Each binder consists of a binder cover, which is a document containing information about the overall content of the work; then you have one main document that stores the most important information relevant to your business case; and finally you have a collection of other documents containing the rest of the information related to this particular business case. All these documents will be passed around together among the people who play a part in solving a business case. The following figure illustrates the structure of a binder:

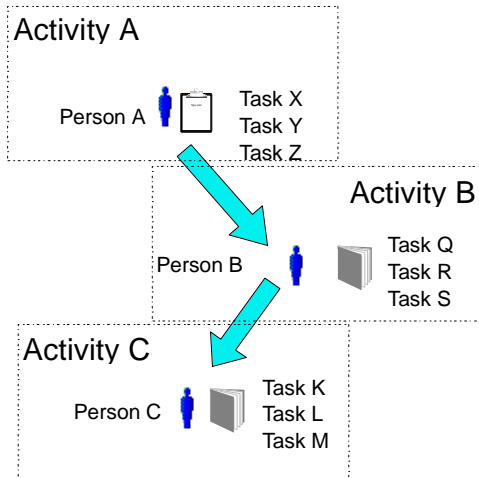


We should also mention that a binder is only a *logical* structure. There is no physical connection between the parts of a binder. Such an approach allows for flexibility when customizing Domino Workflow, as you can show only parts of a binder to your users and refer to its other parts only internally.

Activities and their owners

Every business case has particular people working on it. They are called activity owners, since the binder is their responsibility at some point in time. There is only one person working on a binder at one time (excluding cases of parallel work, which will be discussed later on). Therefore, there is always one activity owner at any given time. As Domino Workflow is people-oriented, so are the activities. All the work that has to be completed by one person before passing the binder to the next person is grouped in one activity. One activity can contain diverse tasks, but as long as one person is in charge of them, they

shouldn't be modeled as separate activities. (There are some circumstances when you might do this, but it's a complex process that involves extending the basic model underlying Domino Workflow.) For example, consider the tasks of reading incoming mail, classifying it, and putting some comments on it. If one person will do the classifying and commenting, both tasks would be part of one activity. If one person classifies the mail and another person comments it, you would model it as two activities.

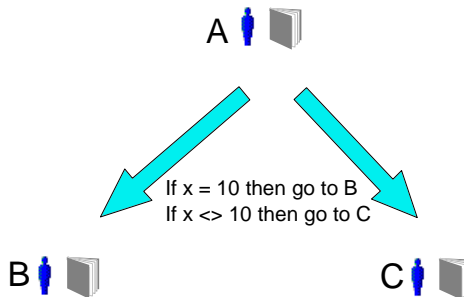


Important Always remember that an activity is a group of tasks related to one person. Splitting your activity into many smaller ones would have a big impact on the performance of your system, since some agents have to be run between activities, and all workflow fields have to be evaluated. Therefore, new activity always implies a new person being in charge of a business case.

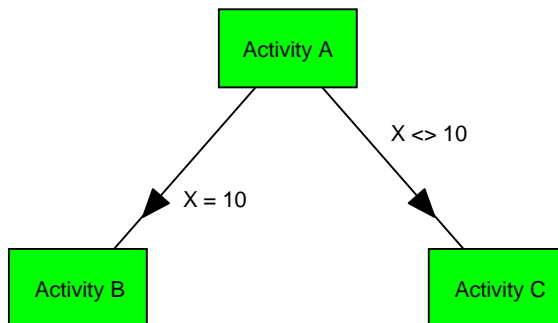
Note Not all the activities have people assigned to them. There are also automated activities included in the model. In automated activities various tasks can be defined that are based on the context and don't require human interaction. Such tasks can include sending an e-mail, executing an agent, or running an executable on the server. The visual representation of automated activities differs from that used for the standard people-oriented activities.

Routing relations: defining the flow

It is typical for most business cases that there are a number of ways of handling them. To create a workflow you have to model all of the ways. Your model must contain not only activities that are used in one particular business case, but also activities that might be used in some other case of the same process. All these activities are related to each other in a specific way, which is described as a routing relation and symbolized through a unidirectional connector that shows who is passing a binder to whom. The following figure shows possible activities involved in one kind of business scenario that are connected through the appropriate routing relations:

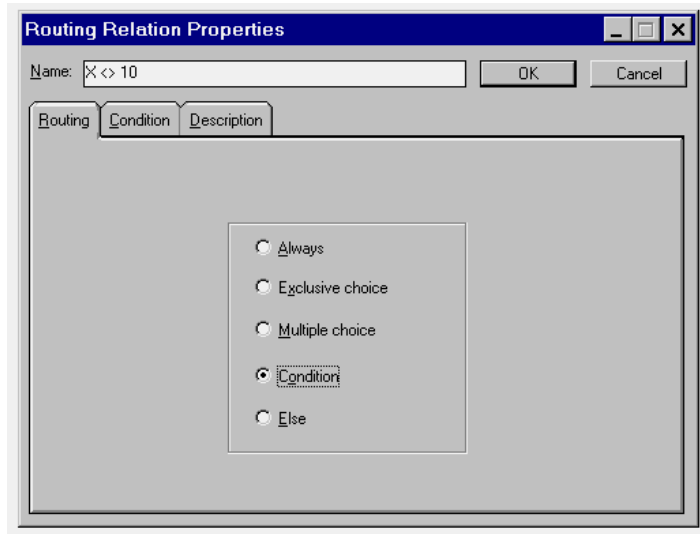


The same scenario can be illustrated using Domino Workflow Architect, which will be introduced later on; the next figure shows this:



Since not all activities will take place in any one particular case, not all the routing relations are the same. Domino Workflow offers you the possibility of defining when each particular way should be taken. You can make this choice by setting the attributes of the connection.

The next figure shows you all the possible attributes of the routing relation:

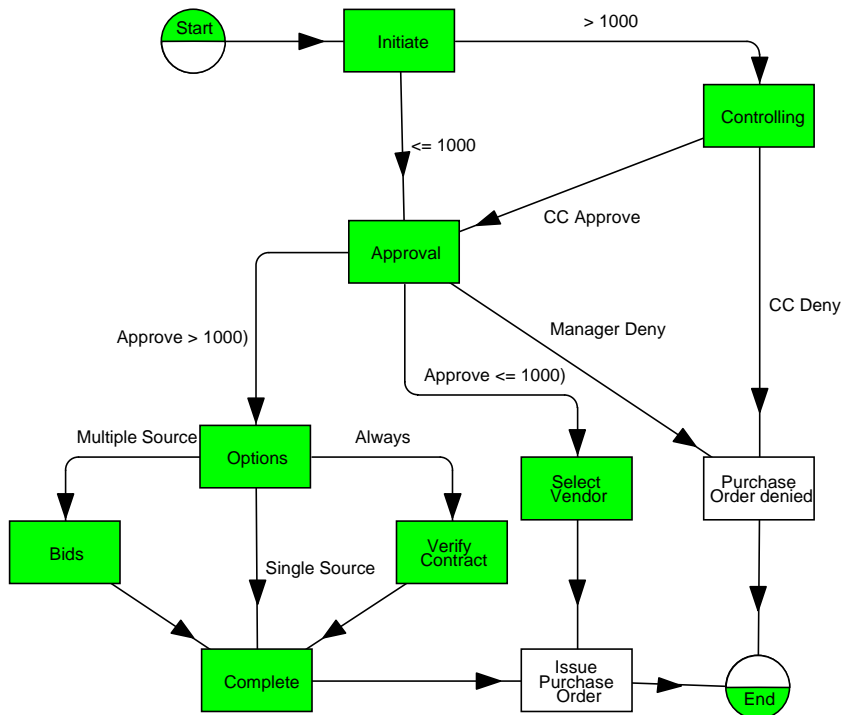


You can decide to forward a binder along a routing relation in any case (Always), you can let the activity owner choose exactly one of the exclusive choice routing relations (Exclusive choice), or you can let this person choose any number of routing options (Multiple choice). There is also the possibility of defining the routing relation without the user's action, based on the job content evaluated through any formula you want to write (Condition). The last available possibility (Else) should always be included in an activity where all the outgoing routing relations are multiple choice or conditions. This routing relation is switched on only if all other routing relations do not fulfill their conditions. Such a connection saves many jobs from producing an error. It is possible to combine any kind of routing relations leaving one activity; they all evaluate independently from each other.

Process map: visual process definition

In Domino Workflow you define your processes visually using a tool called Domino Workflow Architect (henceforth referred to as simply Architect). This tool is used to draw a model of your process, including the start and end points, activities, and routing relations between them. The diagrammatic representation of your process is called a process map.

The following figure shows an example of a process map designed with Domino Workflow Architect. It shows a sample of processing a purchase order; you can find a similar process in the sample database delivered with the product.

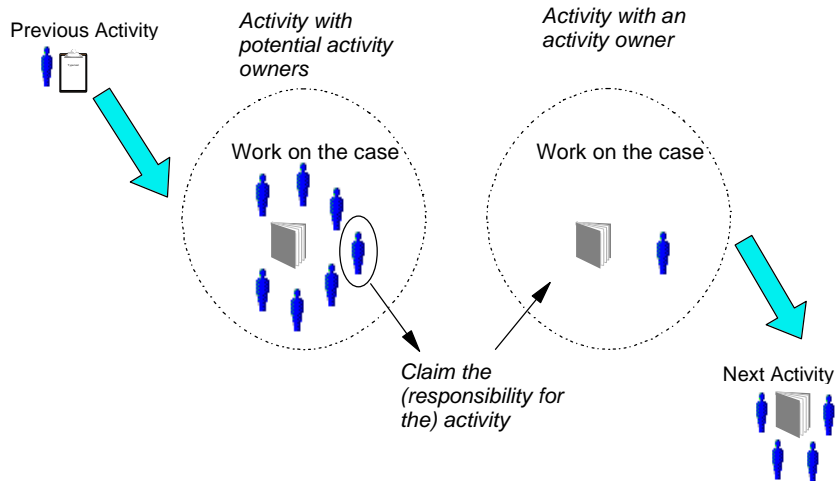


It contains a starting point, an end point, and some activities between them. The activities are connected through various types of routing relations. A process map illustrates exactly the workflow you will use later on, as Domino Workflow will translate its content into a process definition as soon as you activate a process.

Potential activity owners versus activity owners

Creating a general scenario of a business case generally doesn't allow you to define exactly who will be working on each activity. For each activity there will probably be a group of people who could be working on it. It is usually quite impossible to decide in advance which member of such a group will be available and have enough time to work on a specific case. Therefore, in modeling you are not defining the actual activity owners (the individuals who will take the responsibility for an actual case) but potential activity owners — people who could take charge of a case.

Important When designing a process, you always assign potential activity owners to activities. Actual activity owners will have to claim their responsibility for a specific business case. There can be many potential activity owners for one activity but there is always one actual activity owner — the one who claimed the responsibility for an existing activity.



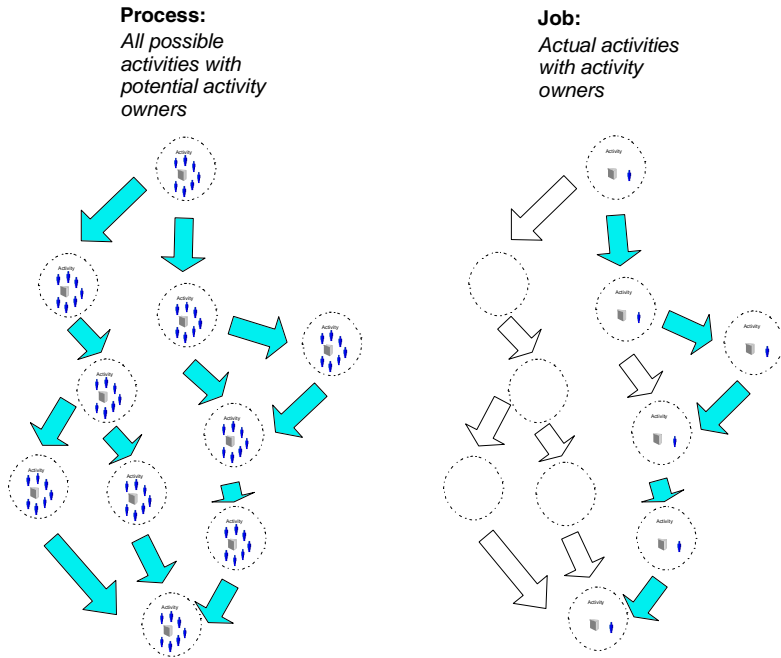
Using groups and roles

When designing your workflow, avoid using specific people as potential activity owners. When you create *groups* and *roles* as participants in your processes, instead of individuals, you can save a lot of work as things change in your company. The process can remain the same, with changes made only to your underlying organization structure and not in the processes themselves. Otherwise, you might end up changing your processes very often, much more often than the logic of your processes changes. Domino Workflow offers you the possibility of defining various kinds of groups in its Organization Directory.

Important Always try to assign roles and groups to your processes instead of using person names. It will save you a lot of effort in maintaining your work.

Processes and their instances: jobs

After you design your activities and potential owners, as well as the documents that will be used in the process, you can activate your process. To *activate a process* means writing its definition on a specific database (Process Definition Database), allowing business cases to be worked on according to the rules you defined. Each business case based on this process is called a *job*. You need to distinguish clearly between processes and jobs. A process contains all the rules and possibilities and a job is a particular instance of such a process revolving around one business case.



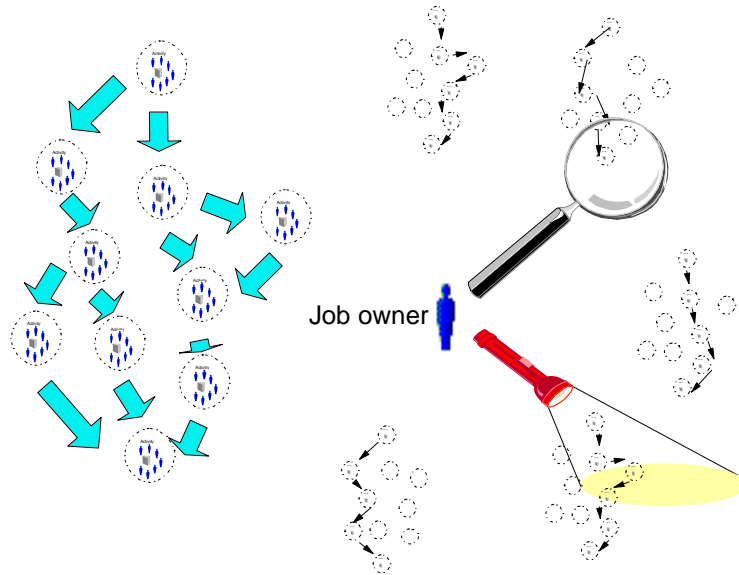
Important A process is an overall abstract definition of all activities that might happen in a business scenario, containing as well the rules of work on all its business cases and potential activity owners. A job is one specific business case, with its data, history, activity owners working on it, and applied rules of handling it. There are multiple jobs based on the same process.

There are three more important concepts that relate to the process models. First is the process owner, the second concerns the way a team works, and the third describes the way parallel paths work.

Managing existing processes: job owners

A job owner is a person or a group of persons that have overall control of the execution of one process; that is, they are responsible for one kind of business scenario. A job owner doesn't always have to decide how things should be handled or necessarily have to participate in defining a process. This is the responsibility of a *process owner* — the person who designs a process in Architect and activates it. The job owner also doesn't have to be involved in every business case, at least as long as everything goes smoothly. On the other hand, job owners have to know the structure of the process very well, since they may be asked to intervene at any point in the process, to take over the job of any people involved in any specific case, or delegate the responsibility. If any errors or unpredictable behavior occurs at run time, the process owners get a notification and are expected to intervene and to

solve the problem. Job owners are administrators of the processes; they should always be looking at the whole picture. You will probably want to assign this function to some of your managers, or at least to the most experienced people involved in any process.

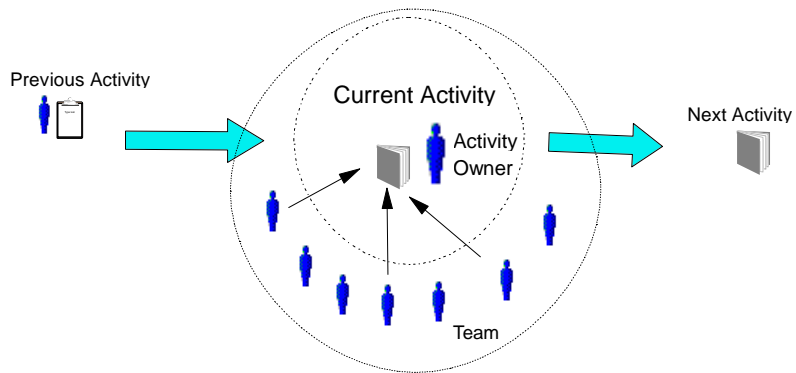


Job owner is set at the time of defining a process and belongs to the process definition. As illustrated in the previous figure, a job owner is supposed to know the process at every stage, and to take a look at any instance of the job at any time something goes wrong that can't be fixed by the current activity owner. It is also the job owner's responsibility to watch over the time restrictions and workload for each workflow participant. Unlike the case with activity owners, who can vary by various jobs, the job owner group stays constant for all the jobs belonging to one process (although it may vary if you change members of the group, which is one more reason to use groups instead of persons when modeling your processes).

Note If you define job owner as a group, it will be resolved to people at the time a new job is created. Therefore, all later changes of the group members don't change the context of the existing jobs. You have to be aware of this in order to check your existing jobs after major organizational changes; there is no automation that will do it for you.

Team concept

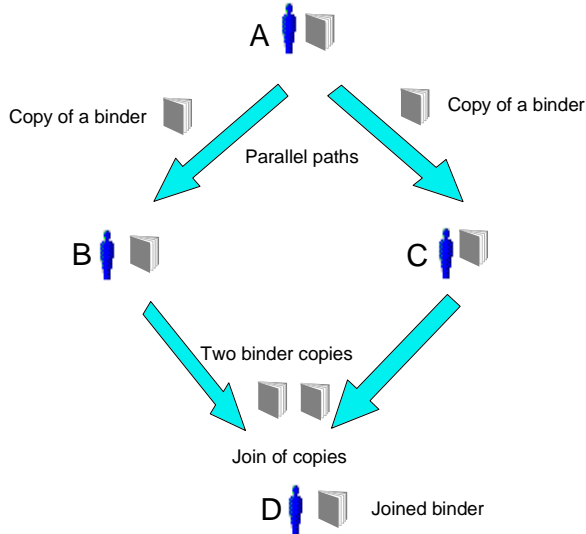
One more concept that has to be introduced at this point is the *team* concept. Everybody knows what a team is: a group of people working together to reach a common goal. In terms of Domino Workflow, a team is a group of people working on one activity. In Domino Workflow there is one special thing about a team: each team has a leader who takes responsibility for this activity, and who is the activity owner of this activity. The activity owner is the only person who can decide when the activity is completed. All the other team members can work on the case, and compose and edit related documents, but they can't decide when the activity is completed because they are not responsible for making this decision. In this definition of a team it will always be possible later on to identify a person responsible for all the actions of the team, also in the situation in which this particular person was only coordinating the actions of the other team members.



Parallel paths: splitting up your binder

When there are multiple paths coming from one activity, there are two possibilities. One is that only one path can be taken for each specific business case, and other paths remain inactive for that case. However, in many situations it is much more effective for several people to simultaneously work on the binder. Domino Workflow allows multiple participants to work on the same document only in the team scenario, where there is an activity owner who can take care of potential save or replication conflicts. All the other scenarios of parallel work are realized using multiple copies of the binder. This allows you to put the parallel paths on different servers or in multiple domains, thereby avoiding most of the potential conflict. The cost of such an approach is the necessity of joining binder copies coming from the parallel paths. There are various strategies for this that you can choose in the advanced process and activity properties in the Architect.

Those settings are described in the product documentation.



Note If you use multiple paths and are sure that there will *never* be two copies of a binder that should be joined, choose the Disable Join property for these activities. This will spare the agents the necessity of checking for the conditions to join the binder and speed up routing. Use this property very carefully, as it can cause a lot of problems if used in the wrong place.

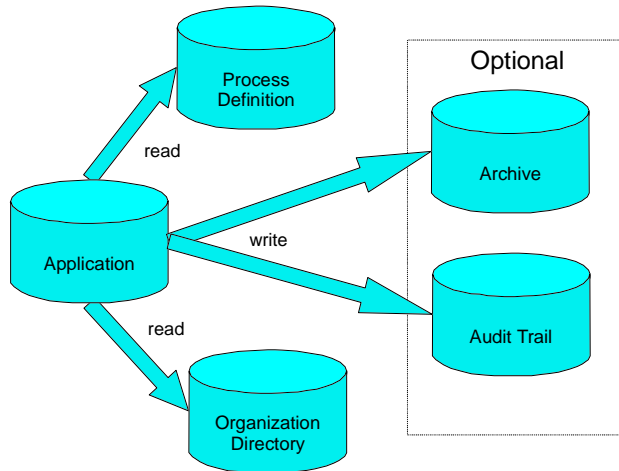
Basic architecture of Domino Workflow

In this section we introduce the main components of Domino Workflow, describe their importance for the whole system, and show how they relate to the other parts of the system. This is intended as an overview of the product architecture; technical discussion and connection details are included in later chapters and in the product documentation.

Working environment: the workflow engine

The main difference between the various parts of Domino Workflow is when they are used. Parts such as Viewer are used only for running jobs based on the process definition; other parts, including Architect and Design Repository, are required to create a process map and process definition. Finally, there are parts that are crucial for getting the job done; these parts are collectively called the workflow engine. The core engine consists of an application database, an organization directory database, and a process definition database. These can be enhanced by adding a separate database for the audit trail and archive.

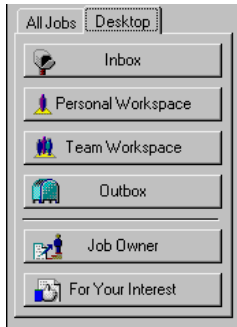
You can see the relationship between these databases in the following figure:



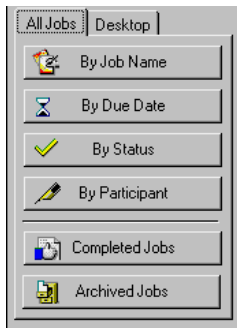
Workflow front-end: an application database

An application database is where the real work takes place. In this database you can store all the information about a business case, check for new work items, and get information about existing work items. It is also where the main part of the workflow takes place. Such an application database can be based on a template delivered with Domino Workflow or it can be any of your existing databases enriched through some workflow elements (this subject will be discussed in detail in Chapter 6). You will probably need some views and elements related to the workflow. You can find detailed descriptions of the standard elements that ship with the product in the product documentation. Among the elements are several views for regular work with the workflow (Inbox for new work items, Personal Workspace for items that you currently work on, Team, and some others), views that are important for people overseeing the business cases (that is, for job owners, who can view jobs sorted on a variety of criteria), and views for the administration of Domino Workflow.

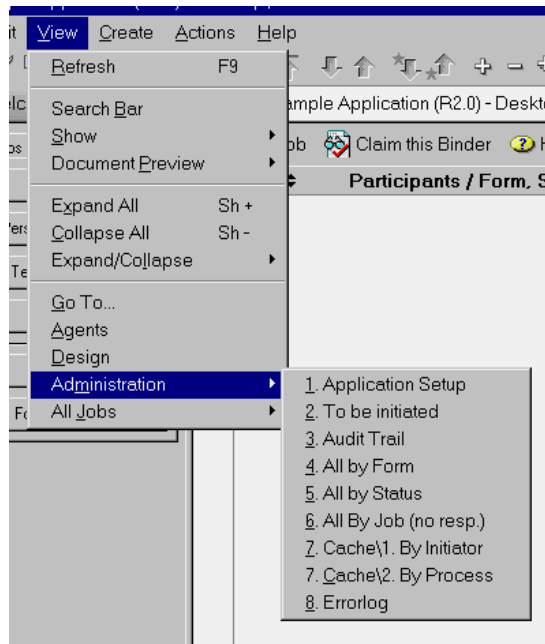
The following figure shows all the views that are designed for the standard work with Domino Workflow:



The next figure shows views built into Domino Workflow that allow the viewing of all existing jobs sorted on some relevant criteria:



Finally, the next figure shows all the views available for administration of Domino Workflow:

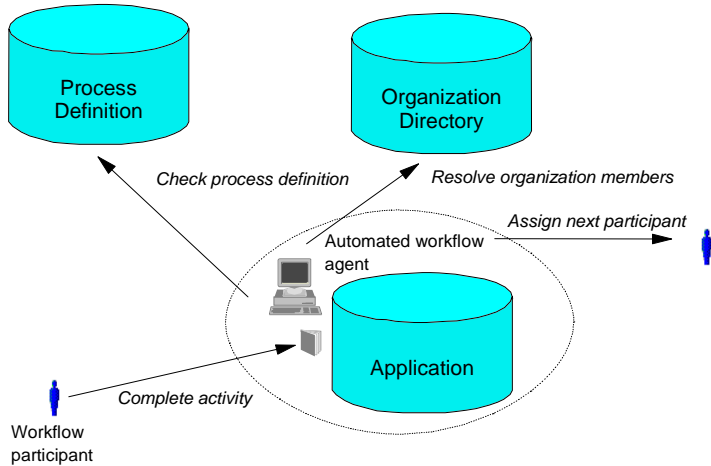


More details about each view can be found in the product documentation. They are mentioned here just to remind you that they exist and to show you where they can be found.

Workflow back-end: an application database

An application database is also the place where agents dispatching work run and the workflow definition is converted to reality. This database is where all the workflow actions get triggered. On the other hand, it can't work alone. Because the data about processes shouldn't be stored in the same place as the data concerning a business case (this is a point on which all standardizing approaches agree), it has to be placed in some other database. This other database, the process definition database, will be discussed later on. There is one more view on processes, and it concerns people and organizational structure. The database that stores such information will be discussed later in this chapter.

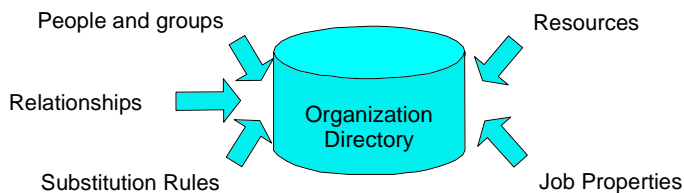
The following figure shows the interaction between the three databases:



Organization model

It is impossible to define workflow without describing its participants. You have to model your organization and the relationships between its parts, as well as the possible points where you might automate the transitions of work. You can do it to some degree using Domino Directory, but you'll very quickly notice that this product is not well suited for such a task. Therefore, Domino Workflow offers you an additional database in which you can model your organization as it is. This database is called the Organization Directory. You can use strictly hierarchical departments, hierarchical workgroups, simple roles, or any combination of all these. Furthermore, you can define any relationship between the elements you want, and you can evaluate a relationship when needed using information from the content of a business case. (An example of such a relationship would be the manager of an employee who creates a request, where an employee would be set at run-time.)

There are also job properties stored in the same database. Job properties are ready-to-use formulas, extracting some of the job context at run-time, that can be used in Architect to design the process logic. (An example of a job property would be an initiator of a particular job.) The database in which all this information is stored is the organization directory. It is a crucial part of the workflow engine.



It requires some effort to synchronize all your people and groups with the Domino Directory, but most of it can be easily automated. There are also ways to eliminate the organization directory, which are described in a later chapter. The better your organization model is, the more you will be able to get from your workflow system. Your goal for introducing workflow is probably to get things done quicker, better, and in a more predictable way. Central to this goal is a good definition of your organization.

Process definition

A process definition database stores the descriptions of all the processes that are currently used. It is a Notes database with regular Notes documents in which information is stored. This is the source of information for the agents stored in the application database that manage the routing of work between workflow participants. You can also find information about the previous versions of the current processes here. The process definitions in this database are created and activated in the Architect, which is discussed in a later section.

Important Domino Workflow always runs a job with the process version with which it was started. Therefore, if you want to delete any old process version from the process definition database, make sure there are no jobs running with this version. There is no automatic way to do it; you have to take care of it manually. The only way to physically delete a process from this database is through the regular Notes mechanisms.

Note You should avoid making changes directly in this database; use Architect for this purpose instead. Furthermore, copying the whole database with processes is not allowed, as Domino Workflow uses internal algorithms using Document Unique IDs. If you copy documents, the document ID will change and can cause routing errors with various error messages.

Archive

As you don't have to use the archive, it doesn't belong to the core of the workflow engine. However, there are standard views provided for your convenience to allow for the quick deployment of the archiving functionality. To use an archive, you must first define it in the organization directory as a resource. The exact steps for doing this are described in the product documentation. You can also use some other kind of archive in your solution. It is only necessary to define an e-mail interface in the organization directory to use the standard mechanisms for archiving provided to you in Domino Workflow.

Audit trail

Audit trail is another database where you can store workflow information. It is generally used for a record of who has done what at what time, and it doesn't contain the actual content of the work. An audit trail database is optional, and therefore it doesn't belong to the core engine. The audit trail information can be stored in the application database also. This possibility can be useful in small databases, where it is not crucial to control their size. To gather audit trail information you have to enable this feature in the Architect, but that's already part of setting up a product anyway.

This concludes our review of the parts of the workflow engine. In the next section we discuss how the engine gets the information on which it performs.

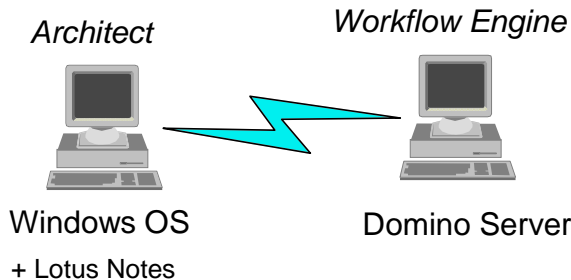
Designing processes

In order to provide the engine with the process definition, you have to create such a definition using special tools provided by Domino Workflow. A discussion of these tools and their relationship to the engine follows.

Visual modeling tool: Architect

The most important tool for defining a process is called Domino Workflow Architect (or just Architect). The Architect is a visual modeling tool that runs under various versions of the Windows operating system. It is basically personal software that can be installed on your PC or installed on a network for multiple users. In either case you'll have to use a Windows operating system on your local PC, as well as a Notes client, to run the Architect. Architect needs your Notes directory as a working directory in order to work.

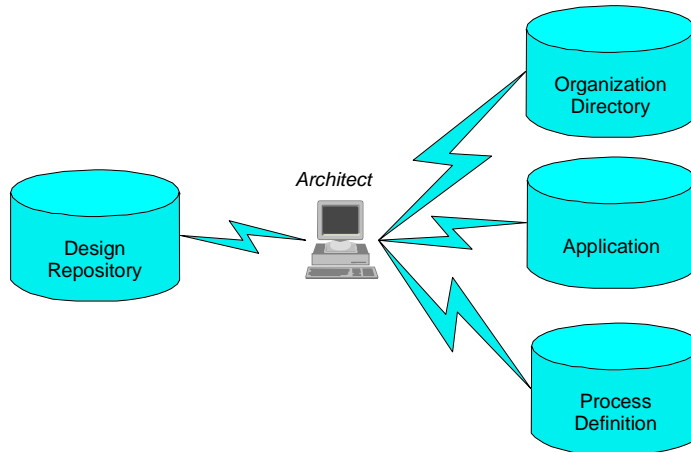
Important Domino Workflow Architect runs on your local machine, *not* on a server. It accesses the Domino server with your identity in order to exchange data.



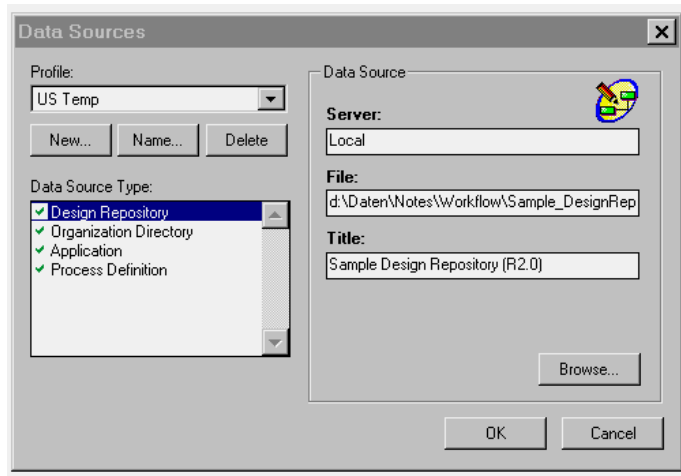
If you try to access any element that requires authentication, the Architect will use your ID together with the password you provide in each session. In this way all the security mechanisms provided by Lotus Domino/Notes can be fully utilized. Architect doesn't change your ID in any way and doesn't include mechanisms to save your password. Therefore, you can consider using it quite safe.

Connecting to the world

In order to define a process you have to connect all the process attributes into one whole construction. This is exactly what happens in Architect. You create a new process in a process window (this is a part of Architect on which you draw a process map by placing activities and connectors and assigning them attributes). You can use objects that were previously designed through the Business Object Library, reuse parts of some other processes that are stored in the Design Repository, or create your own. You will also need some information concerning forms and other design elements from the working environment. Therefore you will need to connect to the application database. On the other hand, to assign people and groups to the activities, it will be necessary that you connect to the organization directory. In the end, when the modeling work is done, you will want to write the information into a process definition database. A connection to this database also has to be set. All these connections have to be set in Architect.



Since there can be various sets of databases you work with, you can define multiple profiles in Architect. The configuration of any group of databases can be set or changed by choosing File - Open databases from the pull-down menu.



Storage facility: design repository

The Design Repository is a Notes database intended to be accessed only through the Architect. It is used to store all of the object information related to process definitions. It also holds information for processes which have not yet been activated or completed. This database is only used for this purpose; you won't find any views here that help you to work with Domino Workflow, and no documents containing user readable information. Use of the Architect is required to make sense of the information stored in this container, or to use and manipulate it for modeling purposes. The existence of this database is crucial for developing workflow, but you don't have to directly monitor its content.

Using all objects: Business Object Library (BOL)

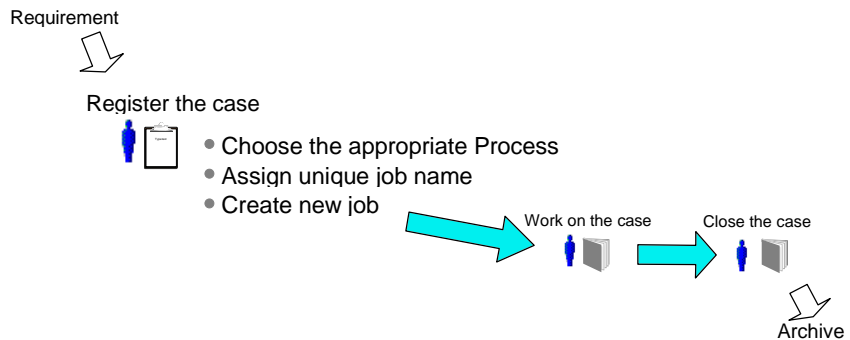
The Business Object Library is a part of the Architect that allows you to utilize all available resources required for modeling a process. It tells you where the resources are located and lets you use them without concern for whether they are stored in the design repository, organization directory, or application database. This tool lets you concentrate on designing your processes instead of looking through various databases for the required elements. Such an approach speeds up your work, boosting your efficiency.

Running your jobs

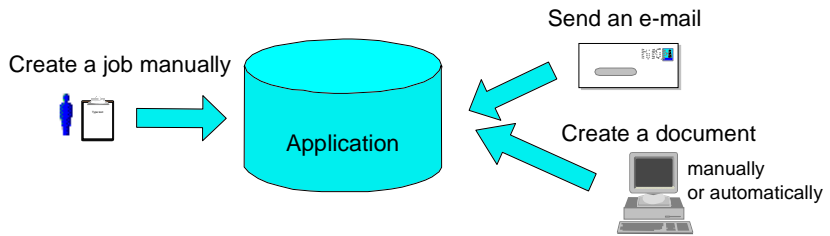
The whole process of setting up the product and defining your workflow is aimed at one goal: to make your work on the business cases more efficient. In this section we discuss how the processes run and what happens when they are done. It is important to understand this in order to manage the processes and to avoid problems. When problems do occur, this information is useful to diagnose and eliminate them.

Initiating a job

In the vocabulary of Domino Workflow, to initiate a new job means to create a new binder that will gather all the information about a single business case and be passed between various people according to the rules defined in some business scenario. Each job has to be assigned to just one scenario. This means that each job has to be assigned to one active process describing what could possibly happen to the newly created binder. Each job also needs a job name to distinguish it from other jobs. The job initiation procedure is illustrated in the following figure, which also specifies the first workflow step:



There are various ways in which a job can be created. The most basic way is using an action provided in the application database that lets you choose among active processes available and define a job name. You can also view the available processes using a Domino Workflow Viewer (referred to simply as the Viewer in the rest of this book), a visual tool that is discussed later in this chapter. You can also initiate a job through an e-mail (your application has to be a mail-in database and this initiation method has to be enabled in the application setup document) or by creating a new document based on a specific form (appropriate settings have to be made in the application setup document and some fields have to be added to the form to enable this method). In addition to being discussed later in this book, all the methods are described in detail in the product documentation.



Working on an activity

As mentioned previously, an activity is a part of a job confined to one person. An activity can be assigned (or offered) to a group of people, but after you have decided to work on one particular activity, you have to claim the responsibility for it. You can do this from the Inbox view or from a binder document. After you have claimed the activity, you are in charge of it — you are an activity owner. It means that you are the only person who can declare this activity as completed. You can allow other people to help you do this work by designating them as team members (if they haven't been assigned to a team through a process default). If you encounter problems in completing the activity you can delegate the responsibility by reassigning it to some other person. Domino Workflow provides a standard reassignment feature; see the product documentation for more details. When you have completed all the tasks that are part of a particular activity, you must specify that it has been completed. Only then will Domino Workflow pass the binder to the next activity described in the process definition.

Routing

Passing the documents over from one activity to the next one is called routing. There are basically two kinds of routing you can use: client routing and server routing. The choice of the appropriate routing option is made during the workflow design process done in Architect.

Client routing is generally preferred for transitions that have to happen immediately. It causes the workflow agents to run the moment an activity is completed. However, parts of the script run on your workstation and you have to wait until they are completed. This waiting time is the main disadvantage of client routing; it must be weighed against the advantage of immediate accessibility for the next workflow participant. Server routing occurs some time after the activity is completed. The agents run on the server according to their schedule. The main advantage of this way of routing is quick reaction to commands like activity complete, but it entails some waiting time between activities.

Domino Workflow needs some information in order to pass the binder. Some of this information is contained in a binder itself, but most of it has to be collected from the other databases, as described in the general architecture part of this chapter. Therefore, after the activity is completed, a process definition has to be recovered from its database, groups and names have to be resolved using information contained in the organization directory, and information contained in the workflow-related fields has to be updated. A successful routing puts the binder into the Inbox of a person or group of people (or changes the status of the binder to Automation in case of automated activity following). If it's impossible to do this, a routing error occurs and a job owner is notified.

Viewing your job

Domino Workflow includes a visual tool for looking at each of your running jobs: the Viewer. This tool gives end-users a better orientation in the context of their job, and provides quick access to the data about a particular job. This tool can be applied only to existing jobs, and only to get some information. The first time a Viewer is accessible is when a job is initiated. At this point all the available processes can be projected visually, along with all their attributes and rules. Later on, the Viewer can show the route a particular business case has taken and provide information from which to derive conclusions concerning remaining time and tasks. To use a Viewer, the option for storing the right information has to be switched on (in the Application Setup document), you will need the role of resource user in the organization directory, and you will need at least reader rights for the binder containing the job you want to view.

Completing the job

There is no special step to complete the whole job; it happens automatically when all copies of the binder created in the job have reached a status of Routing Path Completed. As long as at least one copy of a binder has not reached that status, the whole job is not completed. There might be multiple paths leading to the completion of routing. Multiple paths don't have to be joined explicitly. At the end of routing, Domino Workflow joins all documents into a single binder.

Enhancements to the basic architecture

In this section we describe some ways to use the existing architecture of Domino Workflow in more advanced scenarios. It is not intended as a how-to guide, but instead it introduces some general ideas of what can easily be done and some names of the required features. In this way it should provide you with enough information to understand what has to be done and what features have to be looked up in the help database provided with the product. To get detailed information about this feature, consult the product documentation.

Connecting all the parts

This section deals with the technical realization of the Domino Workflow architecture. This is an overview of the topic; exact details for setting up Domino Workflow are in the next chapter.

Before you can start designing your process it is necessary to set up the databases you'll be working with in Architect. To define a process in Architect you must first define some resources on the Domino side, and then create some forms in the application, where your jobs will be running.

Note You can also use your existing forms, as discussed in Chapter 6.

You will need these forms (or at least some dummy versions of them) to define the basic process and activity forms. If you want to use agents to automate activities, define them before modeling the process; you will not be able to use them otherwise. To assign the right actors to your process you should have a model of your organization in your organization directory. Also, any external resources you want to use should be defined in the organization directory.



The organization directory database contains not only people, groups and relations between them; but also job properties, out-of-office profiles, and resources. In this section we discuss the kinds of resources that can be stored there and used to design processes.

The first kind of a resource is an application database. It has to be defined in the organization directory in order to make routing between multiple applications possible. The definition of the application has to contain its name (in exactly the same form as in the application setup document) and its mail address.

The next kind of resource is a client-driven program that allows you to assign some resources to a job on a process level in the Architect.

Mail addresses allow you to communicate with people or applications not included in your organization directory. Mail addresses are also places on which you can define the location of your archive and audit trail databases in case you want to use them (they have to be declared as mail-in databases in the Domino Directory as well).

The application database also needs some configuration. One application works with exactly one process definition database and one organization directory. Both of those databases have to be specified in the application setup document (accessed by choosing View - Administration - 1. Application Setup from the pull-down menu). In addition, the application database itself must be defined in this database to allow routing between various applications and to enable a Web user interface.

Application Setup Document			
Application Owner Domino Admin/Workshop			
Database settings			
 Organization Directory		 Process Definition	
Server	Sample Organization (R2.0)	Server	Sample Process Definition (R2.0)
Title		Title	
Path	d:\daten\notes\Workflow\Sample_OrganizationDirector_y_20_e.nsf	Path	d:\daten\notes\Workflow\Sample_ProcessDefinition_20_e.nsf
Replica ID	C125678A.00346850	Replica ID	C125678A.0034F6B8
Application			
Resource name			
Server's IP address	127.0.0.1		
Path	C125678A00339745		
URL	http://127.0.0.1/C125678A00339745/OS+WD+A+Name?OpenView		
Global Settings			
Initiation Settings			

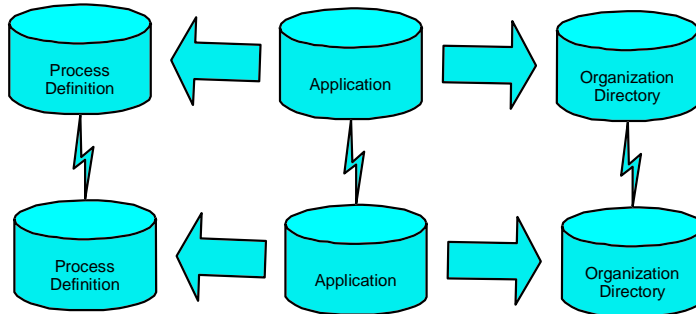
The rest of the components do not have to be specially set up (except for their access control), as they play only a passive role in the workflow and are accessed through other components.

Using multiple applications

As it might not always be appropriate to run all your jobs at all their stages in the same application, there are some other possibilities to consider. This section briefly introduces some of them. The purpose is to show you what is possible without stretching Domino Workflow out of the defined borders. Some of the scenarios are:

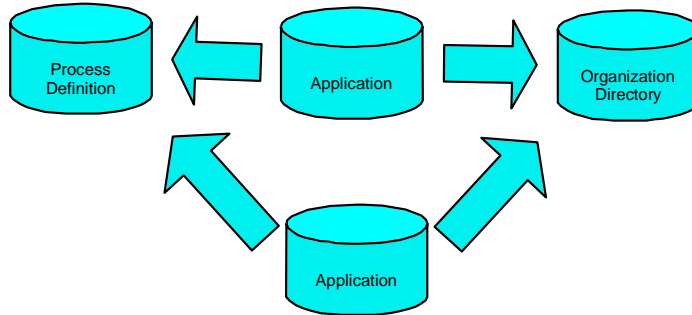
- Distributing your workflow on more than one server through replicas of your application database.

To use this strategy you will also have to replicate the process definition database and organization directory. Be very careful to avoid the possibility of various people taking responsibility for the same binder (claiming it) at the same time on two different servers. You will have to know exactly who works on which server and on each activity to compose its potential owners from people doing their work on the same server.



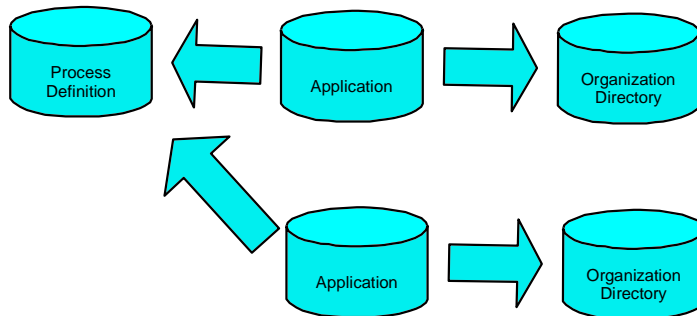
- Multiple applications working with the same process definition database.

This might be an interesting scenario for high volume processes with a lot of jobs. In such a case you might want to have a separate application for each process. You can decide which process will be available in each application by setting an appropriate filter in the setup documents of the applications.



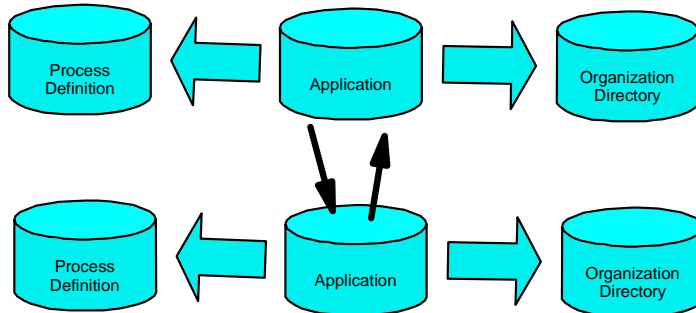
- Multiple applications working with the same process.

You might have the same process running in different parts of the organization, with each part having their own organization directory with the same roles resolving to different people. It is of no importance for this scenario whether the applications are on the same or on different servers.



- Hierarchical processes.

It is possible to substitute a single activity in your process with a whole other process (this is called a *subprocess*). In such a case the second process can run in the same or any other application, using process definitions stored in an appropriate database. Actually, you don't have to know the exact process definition of your subprocess, as it is referenced in the main process only through its name and an application database. Such a construction allows for distributed process development.



There are many other scenarios possible, but we don't want to discuss all the possibilities in this chapter. We are only trying to show some of the standard possibilities which should give you an awareness of the product flexibility and the ability of Domino Workflow to scale up to your needs.

Using alternative organization models

As mentioned previously, it is not necessary to use the organization directory provided by Domino Workflow. You can use Domino Directory instead, or, as a matter of fact, you can use any other organization model Domino can use such as the Lotus Solution Architecture Organization Directory. This allows the use of all the elements offered by the interface without knowing what kind of organization structure is used on the other end.

You should carefully consider whether external directories offer all the information and functionality that is required for your long term workflow needs. We can only challenge you to come up with some relatively complex scenario that you may want to automate in the future, and to check whether you can make a workflow out of it using an alternative organization directory. Make your final decision about which directory to use only after you have done the checking. (You don't have to come up with a working prototype, but you should check the logic of routing against all concepts and architecture).

Summary

In this chapter we discussed some basic characteristics of Domino Workflow. We introduced a few basic concepts underlying this product as well as its basic architecture. You should know by this time what a binder is and what parts it contains, what an activity is and how you should group tasks to design efficient activities. You should also know what an activity owner is and how an activity owner differs from potential activity owners. The concept of a team working on one activity under the leadership of an activity owner should be clear to you. We also hope that you can easily distinguish between an abstract process and its instances, called jobs. The idea of the workflow engine consisting of three databases that don't need any other part to run workflow should also be clear to you. You should know that Architect and its design repository are a separate part of the product required only to define the processes or to change them, working visually. You will need all these basics to understand the following chapters, which discuss some aspects of Domino Workflow in detail. We hope that the process of product configuration will be easier now that you have learned how various parts of it interact. We also hope your overall understanding of the product will let you design better, more efficient processes.

Chapter 3

Installation and basic setup of Domino Workflow

This chapter takes you step-by-step through a very basic, simple installation and setup of the Domino Workflow Release 2.0 product components. The aim is to get you to a state where you can start developing your first Domino Workflow application as quickly as possible.

Many of the advanced and more detailed aspects of setup and configuration are not covered here. Find this information in later chapters and in the product documentation.

This book assumes that you know how to use Notes and Domino as a developer or administrator. If you are new to Notes and Domino, you may want to read this redbook first:

- *Lotus Domino R5.0: A Developer's Handbook*, IBM form number SG24-5331, Lotus part number CC7EDNA

Overview of the setup procedure

The following list gives the required steps for setting up a basic Domino Workflow development environment. Later in this chapter each step is discussed in detail.

The required steps, in the order you should perform them, are:

1. Install the product components on your client workstation.
2. Replicate the database templates and Help database to a Domino server.
3. On the server, create four databases:
 - A design repository database
 - A process definition database
 - An organization directory database
 - An application database
4. Enable and sign agents in the organization directory and application databases.
5. Create and configure setup documents in the organization directory and application databases.

6. Set the Access Control Lists (ACL) for the four databases.
7. Populate the Organization Directory with some Person documents.
Tip Use this list as a checklist to make sure that you did all the steps required for a successful installation.

Before you begin

The minimum requirements to install Domino Workflow are the following:

- The Domino Workflow 2.0 product
You can also use Domino Workflow 2.0, but there will be minor user interface differences. You can download a trial version of Domino Workflow from the Lotus Web site at:
`http://www.lotus.com/workflow`
- A Domino server R4.6.1 or higher for running and testing your application.
- A Windows 95/98/NT/2000 workstation with:
 - Domino Designer R5.0 and Domino Administrator R5.0
Domino Administrator R5.0 comes together with Domino Designer R5.0. You need it to be able to sign database templates. The ability to sign templates is included with Notes Designer for Domino R4.6.
 - or
 - Notes Designer for Domino R4.6

From now on we will refer to either of the development clients as simply Domino Designer.

Domino server requirements

We recommend that you use a “sandbox” server for your initial development of Domino Workflow applications. That way you will not affect your company’s day-to-day work while “playing” with Domino Workflow.

You will need certain access rights on the server. In addition, there are minimum platform requirements.

Your Access Rights on the Server

To create and configure the Domino Workflow databases you must have the following access rights on the Domino Server:

- Ability to create databases.

- Ability to run (at least) restricted agents. To be on the safe side it's better to be able to run unrestricted agents.

If you are not the server administrator, contact that person and ask to have the above permissions set for your Notes ID. This is done in the "Security" section in the Domino Directory server document for your server.

Admin ID

It may also make sense to create an ID for a non-existing user in order to administer Domino Workflow, as recommended in the product. You then use the Domino Workflow Admin ID, instead of your own Notes ID, when creating the databases and enabling the agents. This lets you send all the mail and messages generated by Domino Workflow to the administrator ID, instead of yourself. It also makes it easier to trace the problems generated by Domino Workflow. If you use such an ID, make sure that it has all the required rights, as described above.

From now on we will assume that the Notes ID you are using has the rights to create databases and run unrestricted agents on the server where we are installing.

Installing product components on the workstation

The following components make up the Domino Workflow installation kit:

- Architect program files
The installation process will install the Architect program and put shortcuts in the Windows Start Menu.
 - Engine files
The templates used when creating Domino Workflow databases.
 - Help database
A Domino database containing the online help for Domino Workflow.
 - User guides
PDF files covering topics such as installation, configuration, getting started with Domino Workflow, and the Viewer.
 - Samples
Databases containing sample workflow applications and processes.
- Note** These are *not* the same samples used in this Redbook.

- The Viewer

The installation procedure is straightforward. You start it by running the setup.exe, found either on the product CD or in the directory where you placed the unzipped download files.

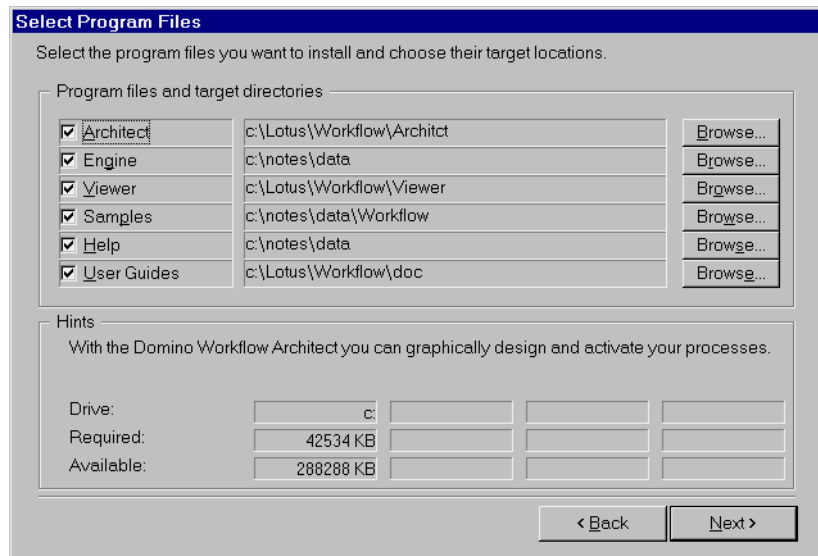
Step through the dialog boxes that ask for your registration information and whether you accept the licensing agreement.

Note A Notes Client must be installed on the machine that you are using for the installation. If there is no Notes Client, the installation program will terminate without installing any of the components.

Note If you have more than one copy of the Notes Client, a dialog box asks which copy you want to use for Domino Workflow. You may choose only one.

Choosing the components to install

Next you are asked to choose what components to install and where to put them. The components are shown in the following figure:



At minimum, select the Architect, Engine and Help components for installation. However, it's a good idea to install all the components now.

You can also change the installation locations for the components. To do so, click on the individual Browse button for a component.

Continue through the other dialog boxes until the installation program finishes.

Creating your work environment on a server

So that various members of your organization are able to develop workflow applications, you should create the workflow templates on a server. To ensure that your local product version reflects the version on the server, we recommend creating replicas of the templates and databases instead of copies. The following list shows the elements that you should replicate to the server:

- The Help database, DWF_Help_21_en.nsf

Note The Help database is installed with replication temporarily disabled. Enable replication on your local copy before replicating to the server. Afterwards, disable replication for both the local and server copies. Databases that don't change, such as help databases, usually have replication turned off.

- The template files:
 - DWF_Application_21_en.ntf
 - DWF_Archive_21_en.ntf
 - DWF_AuditTrail_21_en.ntf
 - DWF_DesignRepository_21_en.ntf
 - DWF_OrganizationDirectory_21_en.ntf
 - DWF_ProcessDefinition_21_en.ntf

Note The design elements in the templates are signed by "Administrator/DominoWorkflow." To avoid problems with Execution Control Lists (ECL), you should sign the design elements in the templates before you replicate.

To see the templates in the template list, place them in the root of your server directory. For the Help database you can choose any place on the server.

Creating Domino Workflow databases on the server

For the minimal initial setup of Domino Workflow, you need to create only four databases out of the possible six types:

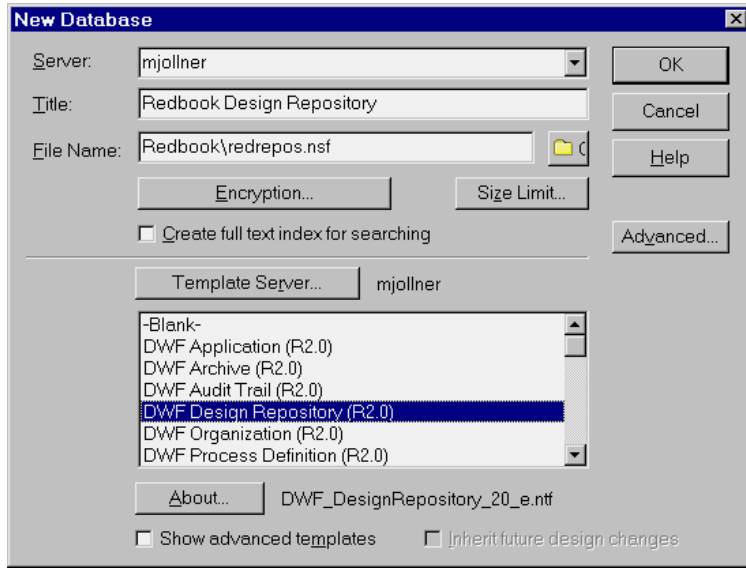
- A Design Repository database
- A Process Definition database
- An Organization Directory database
- An Application database

Create these databases from the Notes Client on your workstation.

General database creation procedure

Use the following steps to create each of the four databases:

1. Select File - Database - New in the Notes Client to get the New Database dialog box (see the following figure).



2. Enter the name of your development server.
3. Enter a title and file name for the database.
Tip Put your databases in a subdirectory so you can find them easily.
4. Click Template Server.
5. Select or type the name of your development server in the dialog box that pops up. Click OK.
6. Select the Domino Workflow template for the kind of database you are creating. For instance, select "DWF Design Repository" when creating a design repository database.
Note All the Domino Workflow templates have DWF as the first part of their title.
7. Click OK to create the database.

Enabling and signing Domino Workflow Engine agents

The Application and Organization Directory databases you just created each have scheduled server agents that need to be enabled. As you enable them, they will be signed with the user ID you use. Therefore, you should have permission to run unrestricted Lotus Script Agents on your development server.

Note In practice, the authors of this book have found that the right to run is only necessary if you run executables as part of automated activities. If your server administrator is cautious about giving you unrestricted agent rights, and you are not planning to run executables in automated activities, you can settle for the lesser right to run restricted agents.

Enabling the agents in the Application database

Open the Agents view in the Application database that you just created. Enable the following scheduled agents:

- OS Domino Workflow Backgrounder
This is the main workflow agent that triggers all the routing actions.
- OS Administration
This agent triggers administrative actions like cache update or archive.
- OS TimeManagement Backgrounder
This agent triggers all actions devoted to keeping track of time settings.

As you enable each agent, you should also specify your development server as the server where the agent runs.

Enabling the agents in the Organization database

Open the Agents view in the organization directory database that you just created. Enable all the scheduled agents:

- Delete temp Forms
- OrganizationCheck
- OS LockCheck
- OS EXPAND CACHE
- OS EXPAND CACHE scheduled

As you enable each agent, you should also specify your development server as the server where the agent runs.

Configuring the Organization Directory database

For the kind of simple, “quick start” setup this chapter describes, there are only a few things you have to do to configure your Organization Directory database.

Later, when you design more advanced workflows or when you place your application into production, you may have to change or add to these settings. For instance, you may have to decide later who should carry the responsibility for the organization directory. Or you may want to configure for pasting in documents from custom databases.

The minimal configuration steps are:

- Create a Setup document.
- Make sure that all the people who will be making any changes in the organization directory are listed in this document.

Caution Only people defined in organization setup as organization owners are allowed to paste, import, or create person documents. Also, only organization owners can change documents. Therefore, we recommend strongly that, *in the introductory phase*, you list all the people who will be evaluating Domino Workflow in the organization directory. However, this setting has to be changed as soon as you switch to production.

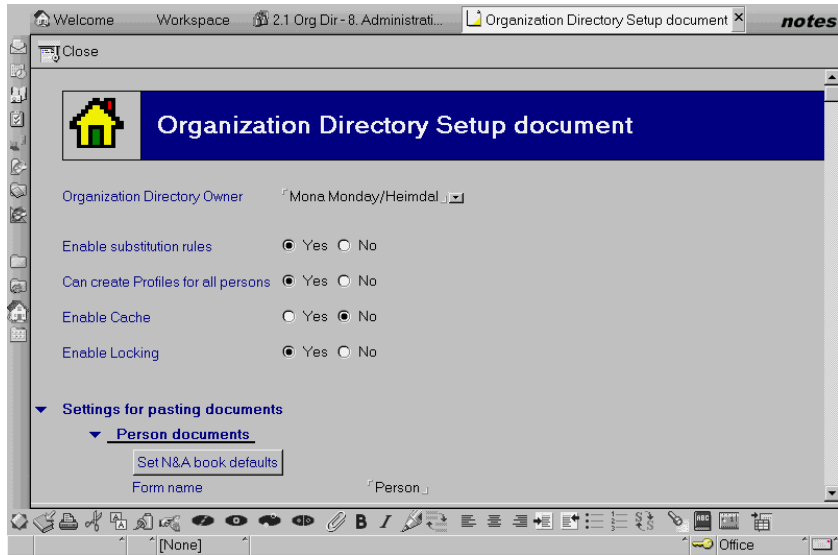
- Additionally, specify settings for pasting documents into the database. It’s not a required action, but we paste person documents from the Domino Directory to populate the organization directory later on in our example.

Creating the setup document

To create the setup document:

1. Open the Administration - Setup view in your Organization Directory database.
2. Click the view action button Create Setup document.

A new setup document opens in edit mode. It will look like the following figure:



Configuring the setup document

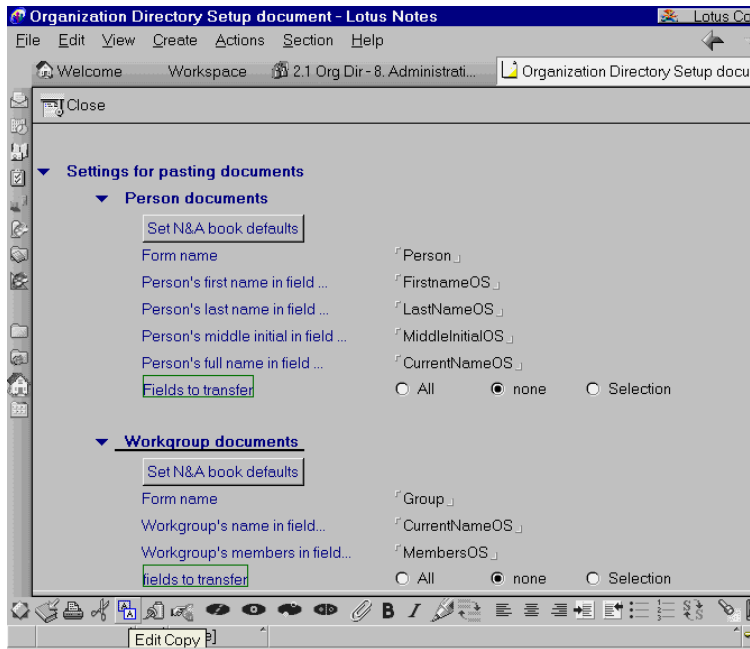
Follow these steps to configure the Organization Directory database:

1. By default, the Organization Directory Owner field contains the current user ID. Add the IDs of all people who will be in charge of the Organization Directory or will want to do any changes in it. Only people listed in the setup document will be able to perform actions such as importing or creating people or group documents and changing the setup document.

If you are using an Admin ID, add your own user ID as well.

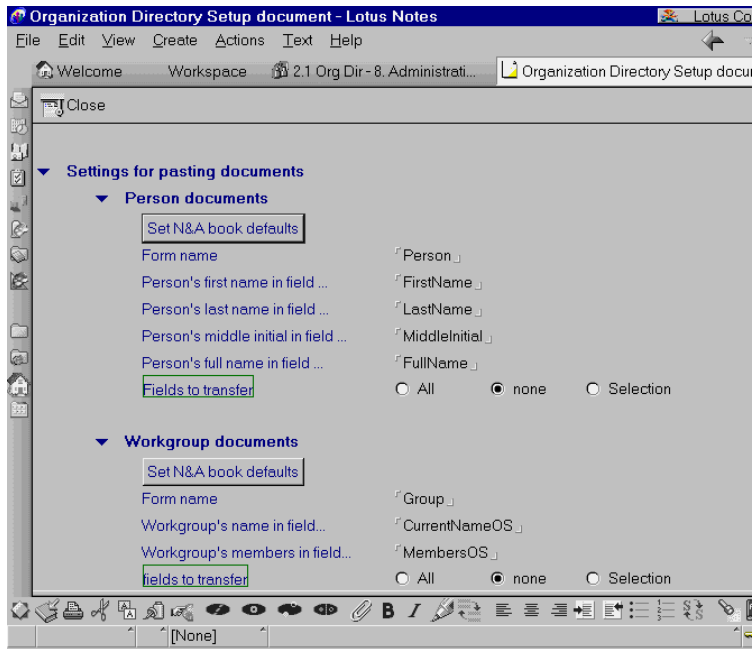
Note Do not delete the Admin ID just yet. You can do so later when you have added yourself to the Access Control List.

- Expand the “Settings for pasting documents” section. Then expand the “Person documents” and “Workgroup documents” sections under that, as shown in the following figure:



- The default paste setup for pasting person and workgroup documents assumes that data will be copied from another Organization Directory database.

Change the paste setup to work with documents coming from a Domino Directory by clicking the buttons that set paste setup to Domino Directory or Notes N&A book. The names in the fields under the buttons will change. For instance, the next figure shows how the “Person documents” fields changes after you click that button.



4. Close and save the setup document.

Configuring the Application database

For the kind of simple, “quick start” setup this chapter describes, there are only a few things you have to do to configure your Application database.

Later, when you design more advanced workflows or when you place your application into production, you may have to change or add to these settings in various sections. Many possible settings are described in the scenarios found in later chapters of this book. You can find information on the rest of these settings in the product documentation.

The minimal configuration steps are:

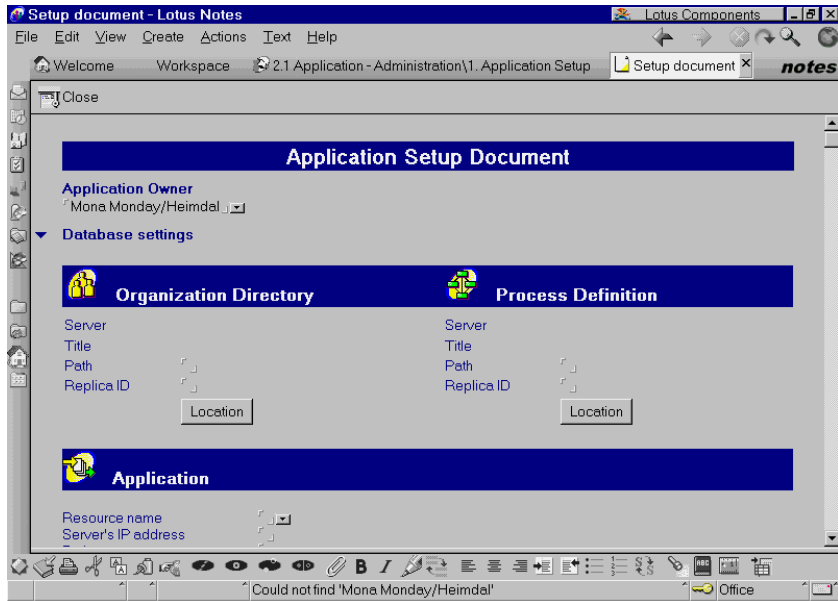
- Create a Setup document.
- Make sure you are named as the owner of database. This will allow you to receive notifications if some errors occur (for instance, those relating to automatic process initiation).
- Specify the Organization Directory and Process Definition databases used with this Application database.

Creating the setup document

To create the setup document:

1. Open the Administration - Setup view in your Application database.
2. Click the view action button Create Setup document.

A new setup document opens in edit mode. It will look like the following figure:



Configuring the setup document

Follow these steps to configure your application database:

1. By default, the Application Owner field contains the current user ID. If that is your ID, just leave it there.

If you are using an Admin ID, add in your own user ID as well.

2. Specify the information for the Organization Directory and Process Definition databases used with your application.

Tip Instead of typing in the database information, use the Location buttons. You then select a path name from the list of databases on your server. When you click OK, the title, path name, and replica ID are set in the setup document.

3. Close and save the setup document.

Setting the Access Control Lists for the databases

You can find full documentation and discussion of Access Control List (ACL) settings for the various Domino Workflow databases in:

- The Domino Workflow Help 2.0 database. For each database type there is:
 - An “About” document that discusses the different kinds of users of that database and the ACL settings for those users.
 - A “How Do I” document that puts the above information into a table.

Tip You can quickly find all these documents under “ACL Settings” in the Index view.

- The *Administrator’s Guide for Configuring the Domino Workflow Engine*. This is an Adobe Acrobat file that comes with the installation kit.

You can find discussion of ACLs in various places in the document. Tables of ACL settings, listed by database, are in Appendix A, “Recommended ACL Settings for the Domino Workflow Engine.”

Quick start ACL settings

After you prototype and test your first few Domino Workflow workflows, you will need to put the above ACL recommendations in place for your defined user groups.

For now, however, you need to make sure that your *own* user ID has “all” rights and roles in the database ACLs. (This is because you will be acting as owner, designer, workflow architect user, and workflow participant.)

For each of the four databases, do the following:

1. Add your *own* full user ID to the ACL if it’s not already there.
2. Give your ID Manager access.
3. For your ID, select *all* roles that are listed.
4. Add your server to the ACL if it’s not already there.
5. Give your server the same access and roles as the “Involved (Domino Workflow) servers” entry.
6. Change the Default to No Access.

Note If you have been using an Admin ID you can switch back to your own Notes ID after changing all the ACLs. From now on you can “be yourself.”

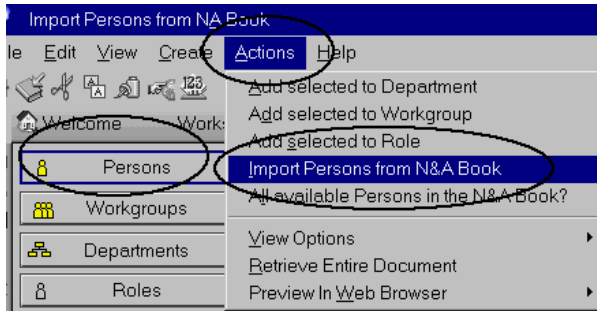
Populating the Organization Directory with person data

This section shows you how to quickly populate your organization directory with person data from the Domino Directory. As you develop and design your application and workflows, you will need to organize Persons into Roles, Workgroups and Departments. Later chapters in this book discuss how to do that in the context of a scenario.

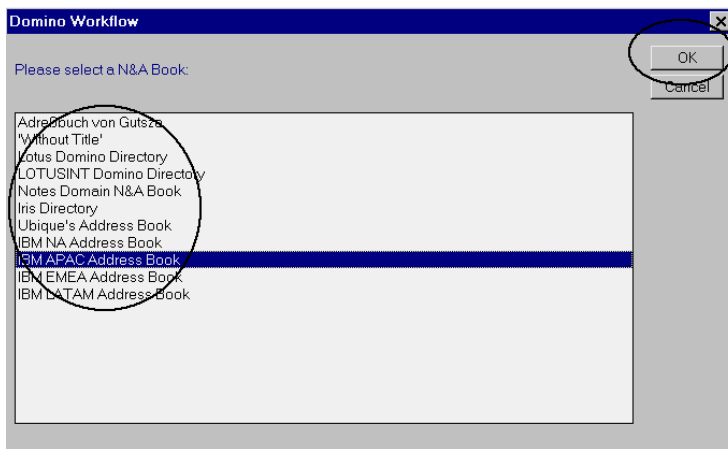
Importing people or groups from Domino Directory

The easiest way to populate your organization directory is by using the import feature of Domino Workflow. You must be an organization owner (according to the organization setup document) to use this feature.

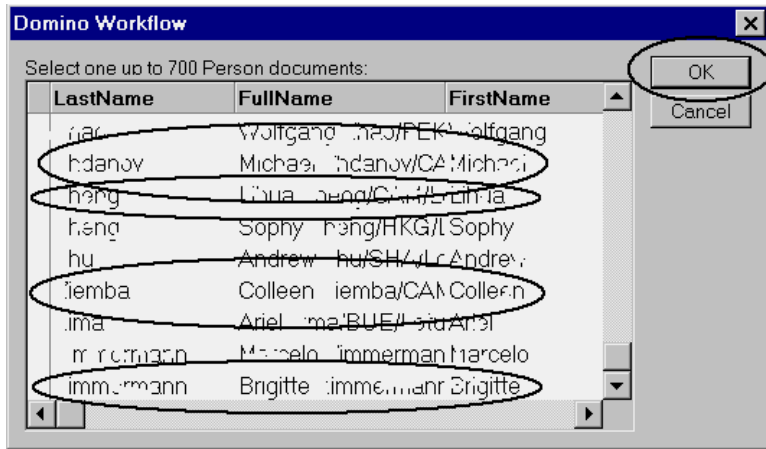
1. Access this feature by selecting the menu item Actions - Import Persons from N&A Book when you are in the "0. Persons" view, as shown in the following figure:



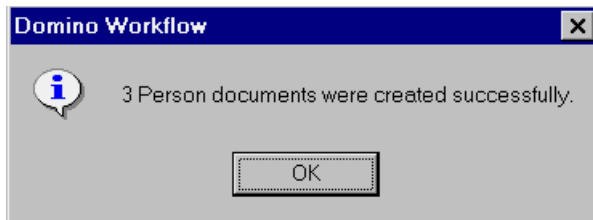
2. Choose the Domino Directory from which your organization members will be imported and commit your choice.



- Choose some names from the Domino Directory. You can choose more than one name by marking each on the left side of the dialog box. In one invocation of the import action you can import a maximum of 700 people into your organization directory. (If you have more people than that, you can do the import action several times, each time importing groups consisting of 700 persons or less). After making your choice, confirm it by clicking OK.



Domino Workflow will confirm the creation of new person documents.



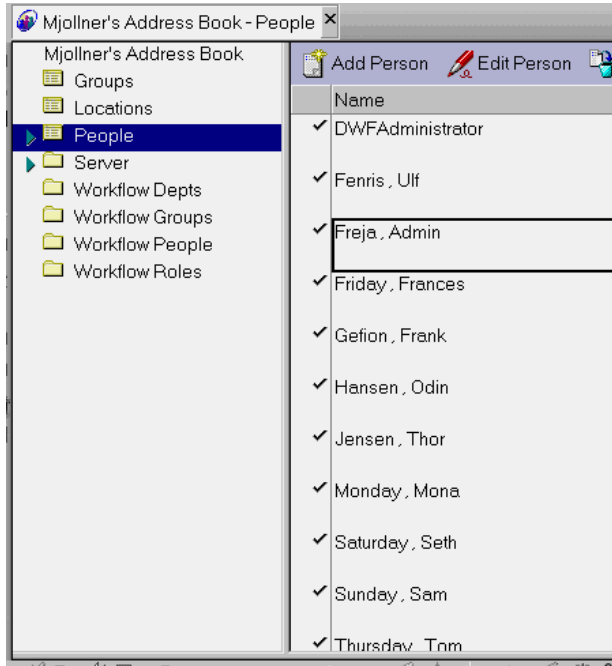
If you try to import people that already exist in your organization directory, Domino Workflow will generate an appropriate message and import only those persons that do not exist yet.

Copy and paste from Domino Directory

Another method to populate your organization directory is by copying and pasting people from the Domino Directory. Use the following steps to do this:

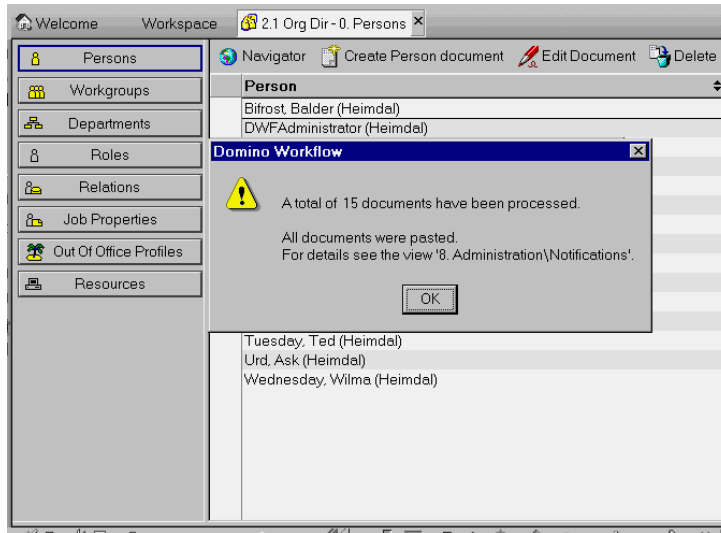
- Set up the Organization Directory for pasting Person documents from a Domino Directory. (See the section “Configuring the Organization Directory database” earlier in this chapter.)

2. Open the People view in the Domino Directory.
3. Select the People you want to put into the Organization, as in the following figure:



4. Select Edit - Copy from the menu.
5. Open the Persons view in the Organization Directory database.

6. Select Edit - Paste from the menu. When the paste is finished you will see a status message, as shown in the next figure:



Review your basic setup

Take time to double check that you have completed all the steps in the basic setup. Did you complete each of the seven steps listed at the beginning of this chapter, in the section "Overview of the setup procedure"? If so, you are now ready to begin designing your first procedure.

Summary

In this chapter we described the process of setting up a simple working environment for Domino Workflow. If you have done everything as described, you'll be able to develop workflow processes using the steps discussed in the next chapter, and you will be able to run your jobs in this newly created environment. We remind you that for real production workflows it will be necessary to make some configuration adjustments, for instance, concerning access rights to the databases. The environment that we created here should only be used to learn how to use Domino Workflow and for trying out its various features.

Chapter 4

Defining your first process

Once you have created and configured a basic set of Domino Workflow databases, as outlined in the previous chapter, you can begin defining your first Domino Workflow process and application.

In this chapter you will learn process design “by example;” the chapter takes you, step by step, through several prototype implementations of a simple workflow process. Each successive prototype will add more complexity to the process, and at the same time introduce one or more features of Domino Workflow.

Only “out of the box” features are used. Mastery of these basic features will give you the capability to design and implement your own nontrivial workflow applications.

What is involved in process design

As with the development of any application, your first step in defining a workflow process is to:

- Understand the elements of the problem to be solved.
- Find out the requirements that a successful solution is to meet.
- Choose the components and tools available for use in the solution.

Obviously, Notes, Domino, and Domino Workflow are among the components and tools that you will use in your solution. But remember, there may be other tools that can be used. For instance, spreadsheet programs and graphic editors can play a part in the solution.

Preparing for the process design: the three Ws

If we say that a Domino Workflow process is the implementation of the answers to the question “Who does what when?”, then a particularly useful way to organize the facts you gather about your problem is to list each of them under one of the “Ws” of that question:

- **Who?** — Who are the people involved in the workflow? What roles do they play? How are they organized? Are the groupings flexible and dynamic? Or more fixed and static?

- **What?** — What is it that the people do? How do they do what they do? Do they approve things? Do they create documents? Edit a part of a document? Call vendors for prices? Plan a campaign? Transfer information to other people?
- **When?** — In what order do people do their tasks. Do they do it sequentially or in parallel? Are the tasks always done, or only sometimes? If only sometimes, under what conditions? How long should each task take?

What is involved in implementing your solution

In general, things involving “Who” mean that you, or an organization administrator, will create entries in the organization directory database.

“What” items usually pertain directly to the application, and mean that you or an application developer will design and implement forms in the Domino Workflow application database.

“When” items are implemented as part of defining a process using the Domino Workflow Architect. As you implement the process in the Architect, changes will be made to the design repository and process definition databases.

When defining a process in the Architect you will have to refer to entries in the organization database, and to forms in the application database. So it’s best to start your implementation with those two databases.

The first prototype: a basic scenario to handle customer requests

The sample problem used throughout this chapter is to design and implement a Domino Workflow application and process that handles customer requests within a company.

This section describes the implementation of the most basic customer request workflow process. However, the first prototype of an application often involves the most upfront work.

In particular, these are the steps you need to do while implementing your first prototype:

1. Organize what you know about the problem into Who, What and When categories.
2. Implement the What portion of the prototype by designing forms in the application database.
3. Implement the Who portion of the prototype by defining the necessary organizational units in the organization directory database.

4. Prototype the solution by defining a workflow process using the Domino Workflow Architect program.
5. Activate and test the prototype process.

Defining the tasks to process a customer request

The following are the individual tasks of the basic customer request scenario:

- Creation of a customer request record. This record needs to contain:
 - Basic information about the customer
 - Details of the request
- Creation of the response to the customer. This involves:
 - Writing the response
 - Editing the response for presentation to the customer.
- Research into the development and marketing issues related to the customer's request, which may affect the response.

Identifying the workflow participants in the basic scenario

When defining processes, you don't want to specify actual individuals as the workflow participants. Instead, you want to identify people by the roles they play in the process, or by the groups they belong to.

These are the organizational groupings and roles that participate in the sample customer request process:

- The people from three of the company's departments participate, in one way or another, in the customer request process. These departments are:
 - The Customer Service department
All the members of this department can be main participants in the customer request process. They are the ones who create a customer request record and subsequently create the response to that request.
 - The Marketing department
Some of the members of this department play a regular and consistent role in researching marketing issues relating to a request. Other members of this department only occasionally assist in request processing.

- The Development department

As with the marketing department, only some members of the development department play a regular and consistent role in researching development issues relating to a request. Other members of this department occasionally assist in request processing.

- The people from Marketing and Development who are regularly involved in request research can be thought of as a “workgroup”. Workgroups are more flexible organization units than departments. A person can belong to more than one workgroup, whereas an individual can only belong to one department.

Although people from different departments can belong to the same workgroup, for the sample problem the workgroups are defined as two separate groups, one for each of the two departments:

- A Marketing Research workgroup
- A Development Research workgroup
- While, as a whole, all members of the Customer Service department participate in customer request processing, there are subsets of people in the department who participate in a more enhanced or in a more limited fashion, or “role”. The roles for the Customer Request process are:
 - The “CS Supervisor” role - people with this role have enhanced capabilities in the process. For instance, Customer Service Supervisors can oversee the work of more junior personnel. They are also the owners of the customer request jobs.
 - The “Junior CSR” role. These are customer service representatives in training. The responses that they write need to be reviewed and approved by a supervisor before being sent on to a customer.
 - Finally, there is the role played by you in the process. Beside designing the application and defining the workflow process, you test it by actually creating jobs and claiming and working on process activities. You have the role of “Testperson.”

Specifying when the basic application tasks are performed

For the first scenario, the workflow model is a simple, basic, temporal ordering of the tasks. This is, in fact, the definition of the three activities of the first process prototype:

1. The Create Request tasks are performed.
2. Marketing and development research tasks are performed.
3. Using any research results from the research activity, the Create Response tasks are performed.

Implementing the “Who”: define your organizational units in the organization directory

Define organizational units used by your process in the Organization Database that you created and configured in the previous chapter. You also need to have previously populated the organization directory with some Person documents.

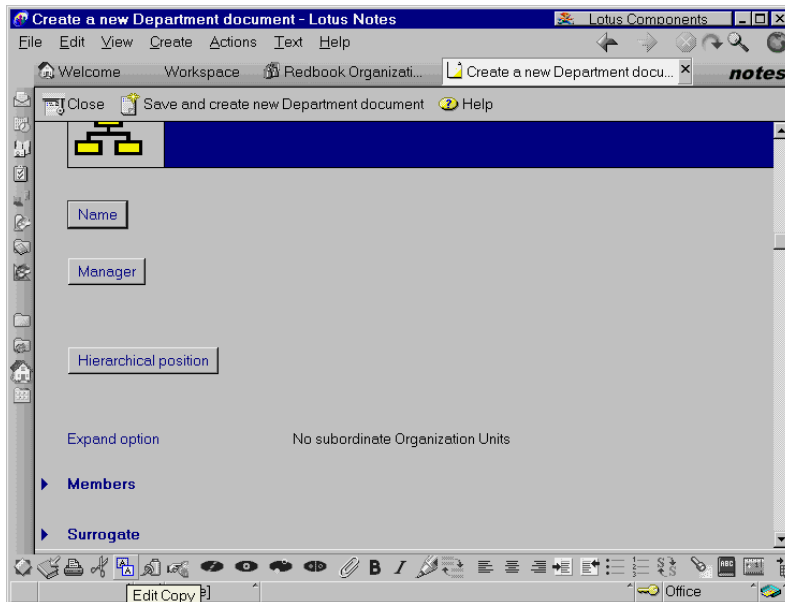
As you define your process using the Architect, you will need to refer to groups or roles. Therefore, you need to create them *before* you define your workflow process.

Create department definitions

For this scenario, you need to create department documents for the Customer Service department, Marketing department, and Development department.

To create a new Department definition, do the following:

1. Open the organization directory in Notes.
2. Click Departments in the left navigator to open the Departments view.
3. Click the Create a Department document action button. You will then see an empty Department document, as in the following figure:

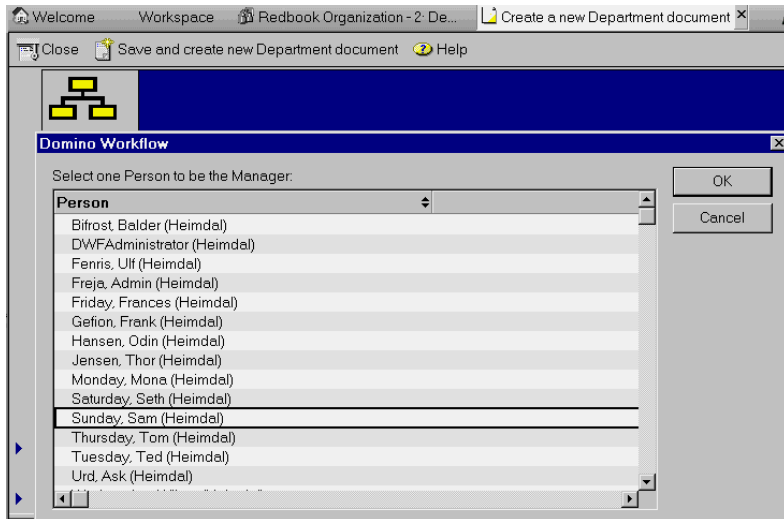


4. Give a name to the Department. Select the Name button and type in the name in the dialog box that appears. We used the following three departments:

- Customer Service
- Development
- Marketing

A department name may contain spaces. However, there are several special characters that are not allowed. The Name entry dialog box lists those invalid characters. Click OK to save the name.

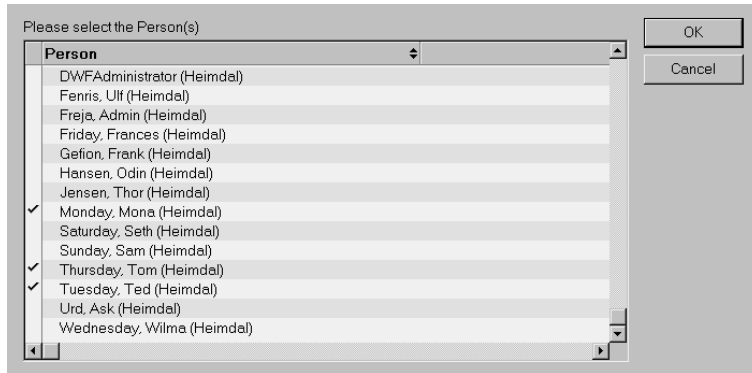
5. Click Manager to specify the manager of the department. A dialog box appears, from which you can pick the name of the person who is the manager of the department. This is shown in the next figure:



You can choose only one person as the manager. Make your choice and click OK.

Note The manager of a department is *not* automatically a member of the department. For instance, if you specify that a department is a potential owner of a job, then the manager is not automatically one of the potential owners. If you want the manager to be considered a member of the department when you refer to the department by name, you must add the manager's name to the Members list in the next step.

- Assign persons as members of the department. Expand the twisty beside Members and then click Add next to the Persons box. You will see a dialog box similar to the following figure:



Select as many people from the list as should be in the department, and click OK when finished.

A person can belong to only a single department. You will get an error notification if you chose a person who is already assigned to a department. You must remove that person from the other department before you can assign him or her to the new department.

- For this scenario we are not specifying surrogates for the department. But if you wish, you can define surrogates.

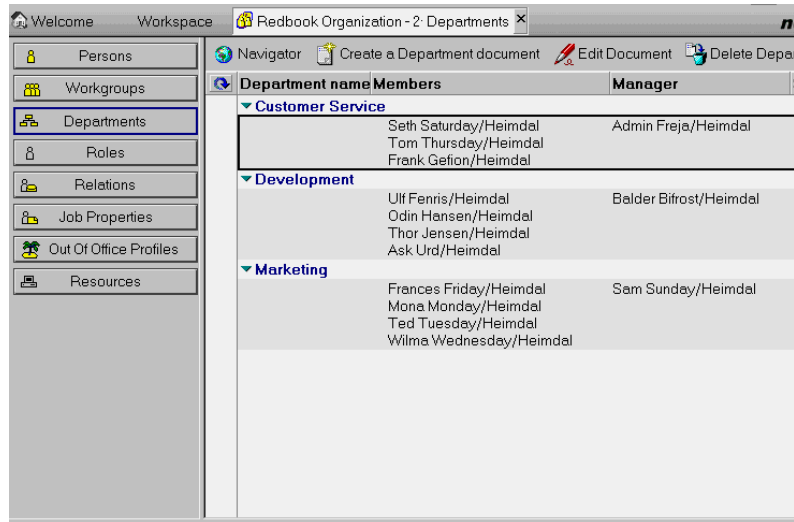
Note Department surrogates are persons (whether specified individually or as part of a group) who can act on behalf of the department when *all* the regular department members are out of the office. A person is considered out of the office when an Out-of-Office Profile document has been created for that person in the organization directory.

- Click “Save and create new Department document” to save the department definition.

Note The new department document does not close after you click Save. It remains open so that you can create other department documents.

- Repeat steps 3 through 7 to create the rest of your department definitions.
- When you are finished creating Department documents, click Close to close the new department window.

You should then see a view of your departments similar to the following figure:



Create workgroup definitions

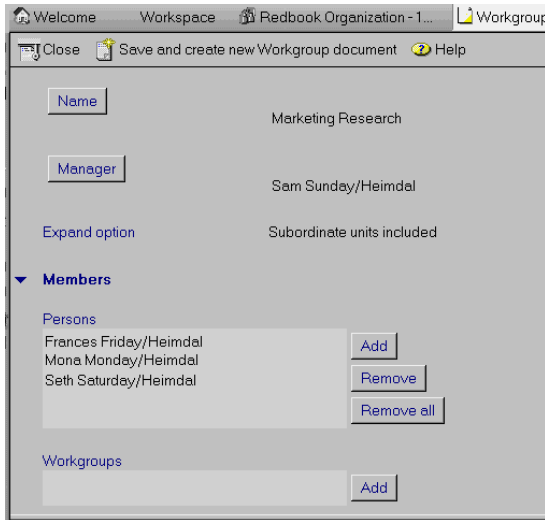
Two workgroups need to be defined for the customer request scenarios: Marketing Research and Development Research. These workgroups define the subset of members from their respective departments that deal with customer requests on a regular basis.

Workgroups are very much like departments, but with these differences:

- A person can be assigned to more than one workgroup.
- A workgroup can contain other workgroups as members.

The steps to create workgroups are similar to those to create departments:

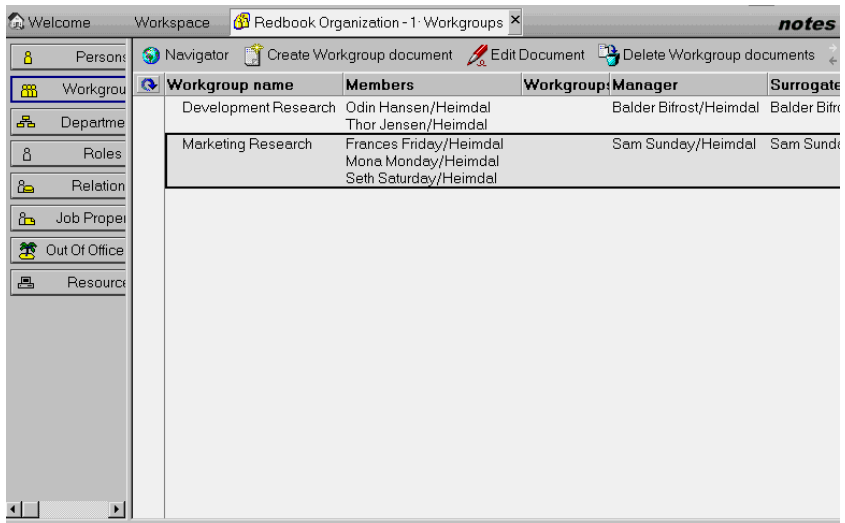
1. Open the Workgroups view by selecting the Workgroups button in the navigator.
2. Open a new workgroup document by clicking the Create a new Workgroup document action button.
3. In the same manner as you did with departments, name the workgroup, specify its manager, and add persons as members. In our example we use the following two workgroups:
 - Development Research
 - Marketing Research



Note For these scenarios we only assign persons to workgroups. If you wish to, you can assign other workgroups to the new one you are defining. In that case use the Add button next to the Workgroups box.

4. Click the action button that lets you save the current document and create a new workgroup document.
5. Repeat steps 3 and 4 for the other workgroup, and then close the new workgroup window.

The workgroups view now looks like the following:



Create role documents

Besides being members of departments and workgroups, individuals can have one or more roles in the organization or workflow processes.

For instance, in our scenarios, there is a role named “CS Supervisor.” People with this role have enhanced parts to play in the workflow process.

You can create roles documents in a similar manner to departments and workgroups:

1. Open the Roles view.
2. Select the “Create Role document” action.
3. Name the role.
4. Add persons as members of the role.
5. Save the role document by clicking Save and create new Role document.

This saves and closes the current role document and opens a new role document. If you have no more roles to add, simply close the new document by clicking Close.

For our scenarios create the following three roles:

- CS Supervisor
- Junior CSR
- Testperson

Important Don’t forget to give yourself the Testperson role.

Another way to add or remove members

You can add or remove a person from a department, workgroup, or role by editing their Person document.

Whenever you make an assignment modification in a Person document, the relevant workgroup, department, and role documents are updated.

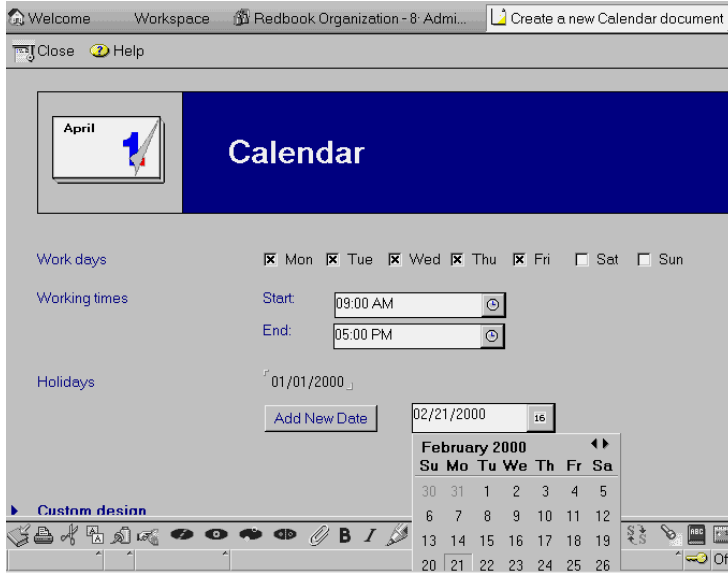
Likewise, member modifications in department, group, and role documents cause updates to the relevant person documents.

Create a calendar document

A calendar document defines information about your organization’s normal working hours and holidays.

The Domino Workflow Engine determines whether an activity is overdue by using the times in the calendar. For instance, if a workflow job is only supposed to take two business days, specified holidays will not be counted as part of the job time.

Access the screen to create or edit the calendar document with the menu selections View -> 8. Administration -> 1. Setup. Use the screen shown in the following figure to specify and change work days and hours, and to add or delete holidays.



Implementing the “What”: design your forms in the application database

Domino Workflow supports the use of multiple forms for the main document. Thus, at each activity, a document can look and behave in a manner tailored to the activity’s tasks.

Note How to specify what form to use in an activity is covered in the sections dealing with process definition. This section only discusses how to create the main document forms.

The basic customer request scenario uses this multiple forms functionality. For the first basic scenario we designed three forms in the application database:

- “Customer Request 1” form used by the persons creating the request. It has the following fields:

Field name	Properties	Value
TicketNumber	Text, Computed when composed	@Unique
CustName	Text, Editable	No default value

continued

<i>Field name</i>	<i>Properties</i>	<i>Value</i>
CustEmail	Text, Editable	No default value
CustAddress	Text, Editable	No default value
CustRequest	Text, Editable	No default value

- A “RequestResearch 1” form used by the researchers. It has the following fields:

<i>Field name</i>	<i>Properties</i>	<i>Value</i>
TicketNumber	Text, Computed when composed	@If(@IsAvailable(TicketNumber); TicketNumber; “”)
CustName	Text, Computed	@If(@IsAvailable(CustName); CustName; “”)
CustRequest	Text, Computed	@If(@IsAvailable(CustRequest); CustRequest; “”)
ResearchComments	Text, Editable	@If(@IsAvailable(ResearchComments); ResearchComments; “”)

- A “ResponseToRequest 1” form used by persons composing the response to the request. It has the following fields:

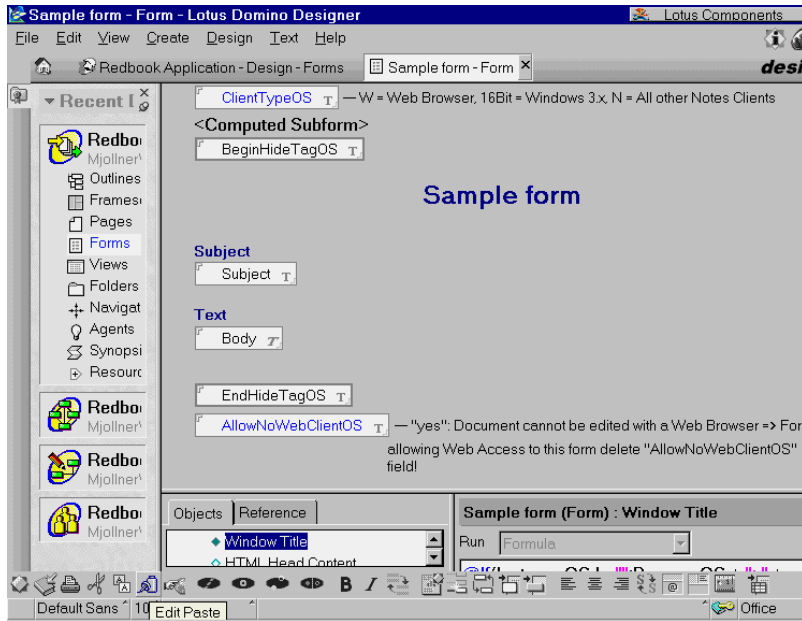
<i>Field name</i>	<i>Properties</i>	<i>Value</i>
TicketNumber	Text, Computed when composed	@If(@IsAvailable(TicketNumber); TicketNumber; “”)
CustName	Text, Computed	@If(@IsAvailable(CustName); CustName; “”)
CustEmail	Text, Computed	@If(@IsAvailable(CustEmail); CustEmail; “”)
CustAddress	Text, Computed	@If(@IsAvailable(CustAddress); CustAddress; “”)
CustRequest	Text, Computed	@If(@IsAvailable(CustRequest); CustRequest; “”)
ResearchComments	Text, Computed	@If(@IsAvailable(ResearchComments); ResearchComments; “”)
RespToCust	Text, Editable	@If(@IsAvailable(RespToCust); RespToCust; “”)

Note All these forms also contain a hidden, computed Subject field. A subject field is required for Domino Workflow views.

Creating your forms using the Domino Workflow sample form

Besides the request-specific fields listed previously, your main document forms must contain fields and design elements used by the Domino Workflow Engine.

An easy way to ensure that you have the required workflow fields is to base your forms on the sample form that comes with the application database.



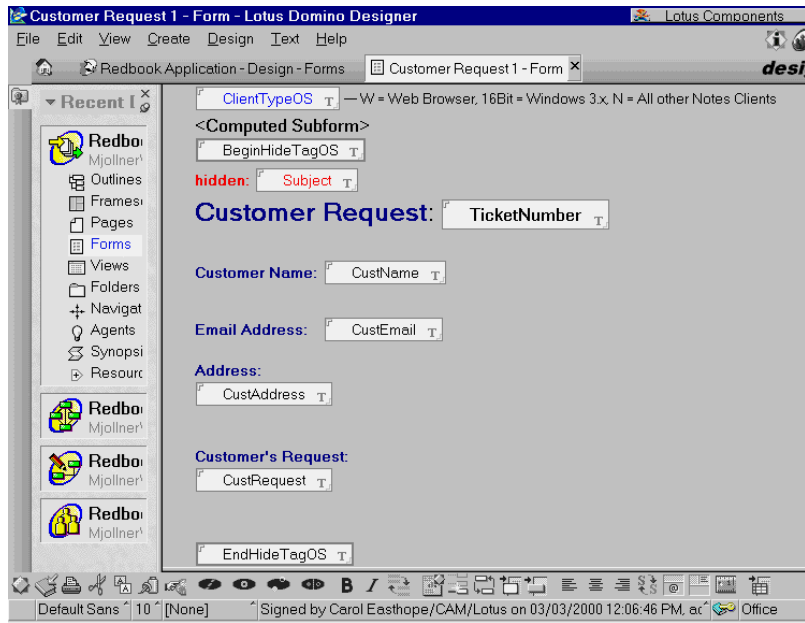
Using the following procedure, create the “Customer Request 1”, “RequestResearch 1”, and “ResponseToRequest 1” forms.

To create your forms:

1. In Domino Designer, make a copy of the application database form “Sample form.” (See the previous figure.)
2. Open the copy and name it.
3. Place your application-specific design elements between the existing fields BeginHideTagOS and EndHideTagOS. Do not change anything before or after those fields. You can modify the look and feel of the form as you wish. You can also delete the Body field.

Important The form must contain a text field named Subject. (It’s used like a title for the document in the various workflow views.) However, it need not be a visible or editable field.

The figure below shows the “Customer Request 1” form used in the first prototype:



4. When you have finished making your design changes, close the form and save it.

First prototype continued: implementing the “When” using the Architect

A Domino Workflow process is what ties the “who” and “what” together in a “when” sequence. Before beginning to implement the process, you need to make the changes to the organization directory and the application database as described in the previous section.

The steps to create your first process in the Domino Workflow Architect are:

1. Create a database profile and open all the databases you need for your application.
2. Create a new process and set its properties.
3. Create activities in the process, and set their properties.
4. Connect the activities by routing relations (arrows).
5. Check process syntax, and correct if necessary.
6. Activate and save the process.

Creating a database profile and opening the databases

All process and activity definition takes place in the context of the currently opened design repository, process definition, organization directory, and application databases.

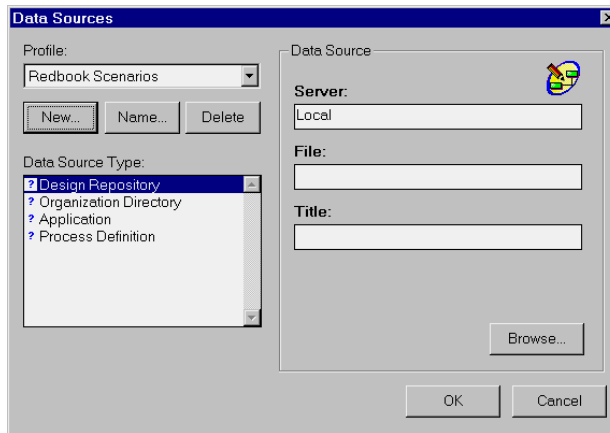
Therefore, after you start the Architect but before you start defining a process, you need to open all four of the Domino Workflow databases.

Caution The Architect always starts up with the database profile that was last used. If you chose to install the sample databases, the Architect was configured with the “Sample” database profile as its “last used profile.” When you start the Architect for the first time, you have the sample databases as your context.

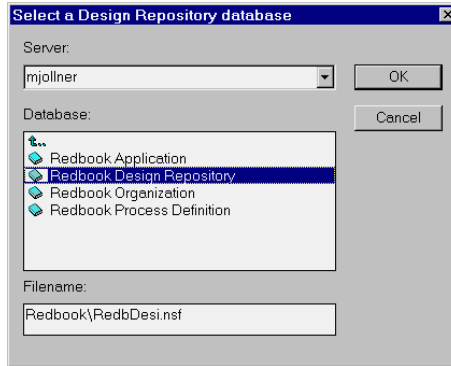
All your databases can be opened at once if you have defined a database profile for the set. To define a profile and open the databases:

1. Choose File - Open databases from the menu.
2. A Data Sources dialog box opens. Click New. Then, in the Profile Name dialog box, type in a name for your database profile. For instance, enter **Redbook scenarios** as the name. Click OK.

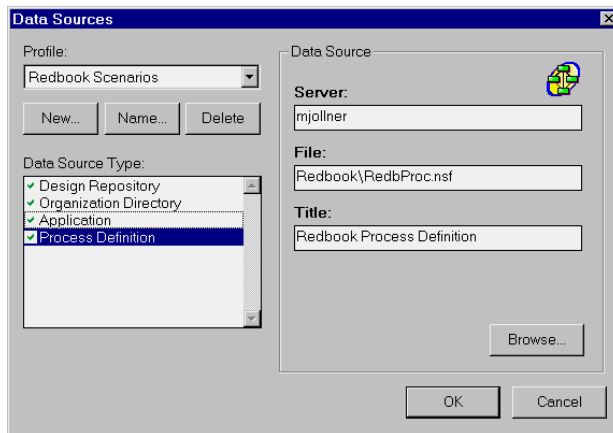
The Data Sources dialog box now looks like this figure:



3. Select a type in the Data Source Type list and click Browse. In the database selection dialog box, choose the server location of your database and then choose the database, as in the following figure:



4. Repeat step 3 for each of the database types. A check mark is placed before each database type if you have specified a valid database for that type. When you are done, the Data Source dialog box looks like the following figure:



5. Save the profile and open all the databases by clicking OK.

Tip If you work in the Architect with more than one application, define a database profile for each one. You can then quickly switch from one set of databases to another.

Creating a new process

You can create a new process by choosing File - New Process from the menu.

If there is not an open toolbox window, one now opens. The default toolbox location is the left side of the Architect. You can change the location of the toolbox from the Options menu.

An empty process window also opens, ready for you to start drawing your process.

However, it's a good idea to set the process properties before you begin defining activities. There are two categories of process properties: basic and advanced. For the first prototype, you only need to specify basic properties. (Advanced properties are discussed in later prototypes.)

Setting the basic process properties

You can access the basic properties screen by either of the following methods:

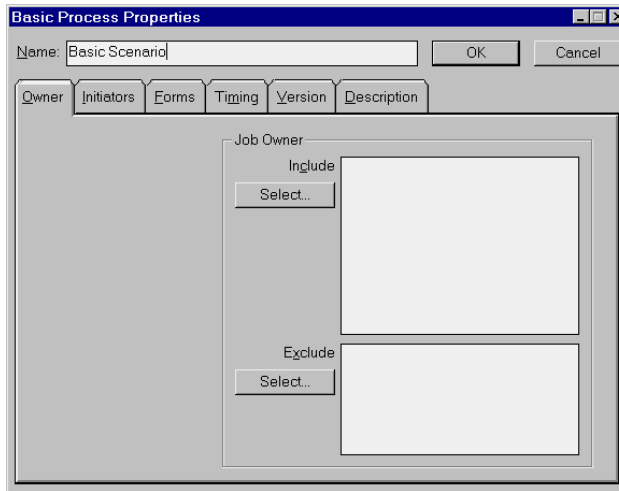
- Select Process-Basic Properties from the Architect's menu bar.
- Right-click on the empty space in the process window, then select Basic Properties from the pop-up menu.

Either way, the same Basic Process Properties dialog comes up. Enter a name for your process. In the first prototype we use the process name

Basic Scenario

Tip For a production process, give the process a short, descriptive name that the end-users understand.

The basic properties dialog box is a tabbed one that looks like the figure below. The next subsections discuss each of the tabs in turn.



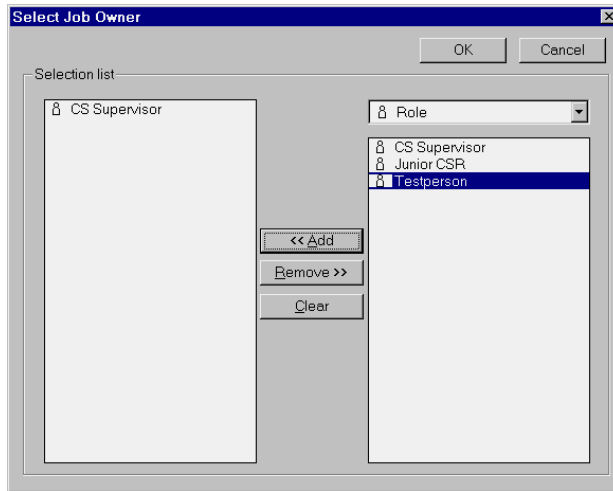
Job owner property

Job owners are responsible for the overall execution of a process. Compared to Activity Owners, who are responsible for only one activity, job owners have wide-ranging rights to intervene in jobs. Job owners play a more managerial than technical role in a job.

Specify who the job owners are by clicking the Include Select button.

To exclude persons or groups as job owners, click the Exclude Select button. You would use Exclude if, for instance, you want to include a department as a job owner, but exclude the more junior members of the department from job ownership.

For the customer request scenarios, we assign persons with the CS Supervisor role as the job owner, as show in the next figure:

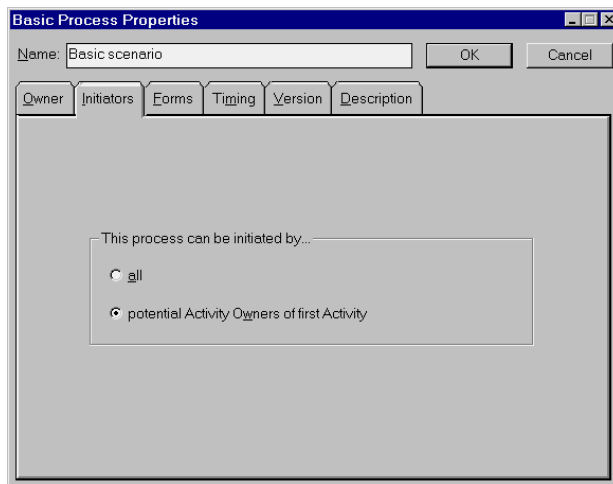


Important Never assign just one individual as a job owner. Once a job is started its actual owner or owners do not change. If only a single person is specified as a job owner, and that person becomes unavailable, then no one, not even the surrogate, can act as the job owner.

Click the OK button to close the Job Owner property dialog when you have made all your selections.

Process Initiators property

This property specifies who may start a job. The choices are depicted in the following figure:



All means anyone with access to the application database can initiate a job.

Potential Activity Owner of first Activity means only those who are potential owners of the first activity in the process can initiate a new job. This is the default setting.

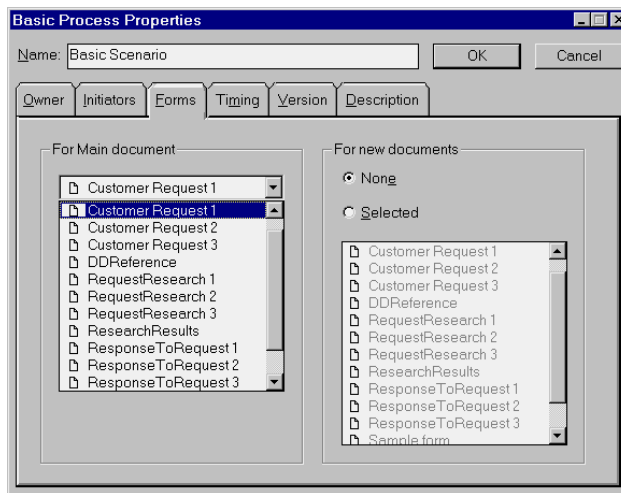
For the customer request application only members of the Customer Service department can initiate jobs. Therefore, we accept the default setting for the Initiators property, as we will assign that department as the owners of the first activity.

Process forms property

This tab is where you specify the form to use for the main document in the job's binder. You can also specify if any other documents in the binder are allowed, and if so, what forms to use for them. These settings can be considered the "process defaults" since you can define a forms property for each activity.

For the main document, select the Customer Request 1 form that you designed in your application database.

Also, select the None radio button under the "For new documents" section, since this scenario does not use any other documents. The Forms tab now looks like the following figure:



Process timing and version properties

The Timing tab is used to specify how long a process should take. You can also specify whether job overdue notifications are sent to the job owners, and when. By default nothing is set in this tab. This chapter's scenarios do not use process timings.

The Version property is set and updated by the Architect program every time you save the process. You cannot modify the property directly. The Version tab shows the process version number, who created and modified the process, and when.

Process description property

The Description tab is used to enter a short description of what the process does.

This is displayed to end-users when they initiate a new job. It helps them to make the correct choice among the available processes.

The process description is also put in the job's cover document.

Click the OK button to close the basic process properties dialog when you have entered your information.

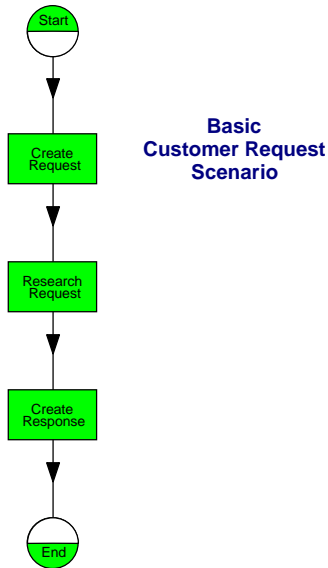
Creating and defining the process activities

Only basic routing is used in the first scenario, Basic Customer Request. This means that there are no special routing situations. Each activity follows sequentially after the other, from the start of the process to the end.

Tip Drawing in the Architect is easy and intuitive. However, you may want to practice drawing a bit before you start drawing a process diagram in earnest. For in-depth instruction on drawing process diagrams, see the Domino Workflow Help database.

Using the Architect's drawing tools, construct a three activity process diagram that has the same shape as the scenario diagram in the next figure.

Note Your activity boxes won't have names until you set the activity's properties.



Accessing activity properties

Now you are ready to give names to your activities, and set their other properties.

Like processes, activities have two types of properties: basic and advanced. To access an activity's properties you can do one of the following:

- To access the Basic Properties set, double-click on the activity.
- To access either the Basic or the Advanced Properties set, first select the activity box. Then click mouse button two (normally your right button) once. You will get a pop-up menu where you can select either of the property sets.

Defining the create request activity

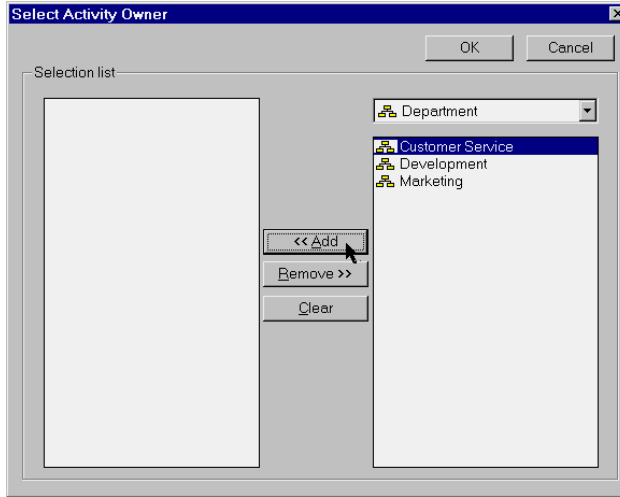
The first activity in the process is Create Request. To name it and define it further:

1. Select the first activity box in the diagram and access its basic property set.
2. Type Create Request in the Name field of the properties dialog box.

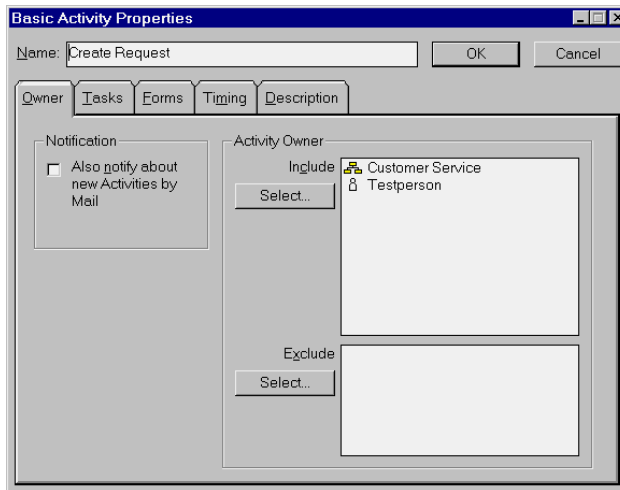
Activity owner property

Now select who may be an owner of this activity.

For all scenarios in this chapter, only members of the Customer Service department may initiate and create a customer request. So choose that department, as in the following figure, as the activity owner:



Also select the Testperson role as a potential activity owner and click OK to close the dialog box for selecting the activity owner. The Create Request owner tab now looks like the following figure:



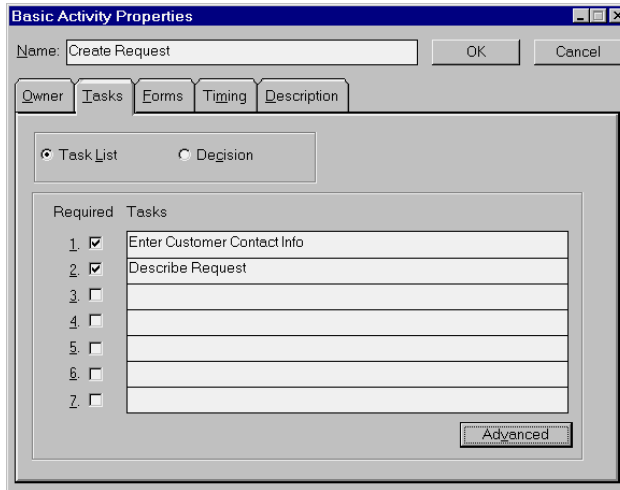
Activity tasks property

The Tasks tab lists all the tasks that make up the Create Request activity. We have two required tasks defined for the Basic Scenario:

- Enter Customer Contact Info
- Describe Request

Click on the Tasks tab and write the tasks in the dialog. Set a mark in the check box for each task to make them required.

You can see the tasks in the following figure:



The screenshot shows the 'Basic Activity Properties' dialog box with the 'Tasks' tab selected. The 'Name' field contains 'Create Request'. The 'Task List' radio button is selected. The 'Tasks' table is as follows:

Required	Tasks
1. <input checked="" type="checkbox"/>	Enter Customer Contact Info
2. <input checked="" type="checkbox"/>	Describe Request
3. <input type="checkbox"/>	
4. <input type="checkbox"/>	
5. <input type="checkbox"/>	
6. <input type="checkbox"/>	
7. <input type="checkbox"/>	

An 'Advanced' button is located at the bottom right of the dialog.

The tasks entered here will show up on a task list on the cover document and the user can check them off as they are completed. Domino Workflow does not allow routing to the next activity before all required tasks are marked as completed.

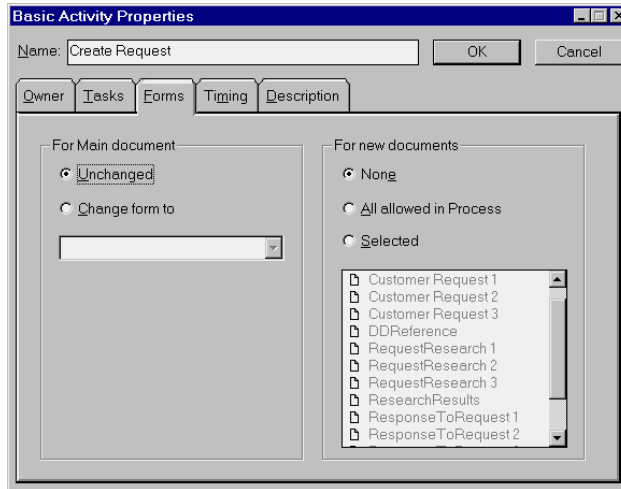
The Decision button is discussed in a later scenario.

The Advanced button is used to specify resources and tools available to the end-users when they perform the tasks. See the Domino Workflow Help database for more details on this feature. It is not used in any scenarios in this chapter.

Activity forms property

The Create Request activity uses the same form for the Main document that was defined for the process, so you select the Unchanged radio button on the Forms tab (see the next figure).

Also, in the “For New documents” section, select the None radio button.



Activity timing and description properties

The Timing tab is the place to specify how long an activity should take, and what notifications should be sent for overdue and unclaimed activities. The scenarios in this chapter do not use activity timing. However, in real world processes, you should use process and activity timing features.

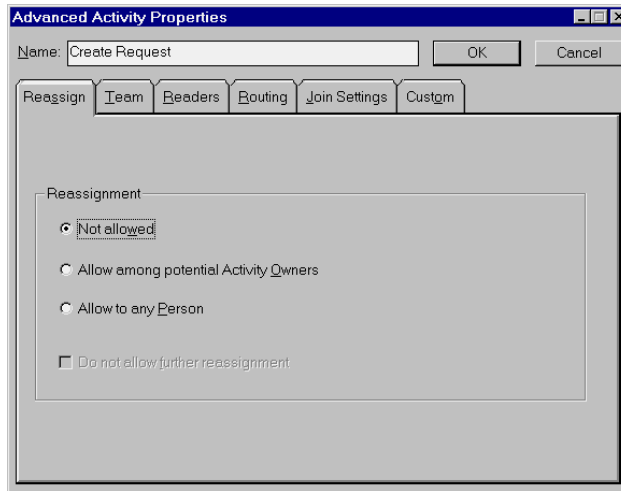
The activity’s description is placed in the cover document. Enter a brief description of the activity, or instructions for the activity owners.

Click the OK button to close the basic properties dialog when you are done.

Activity reassign property (advanced)

When handling customer requests we want the same person who initiates the job to also complete the first activity. To enforce this, set the Reassign property to “Not Allowed,” as in the following figure:

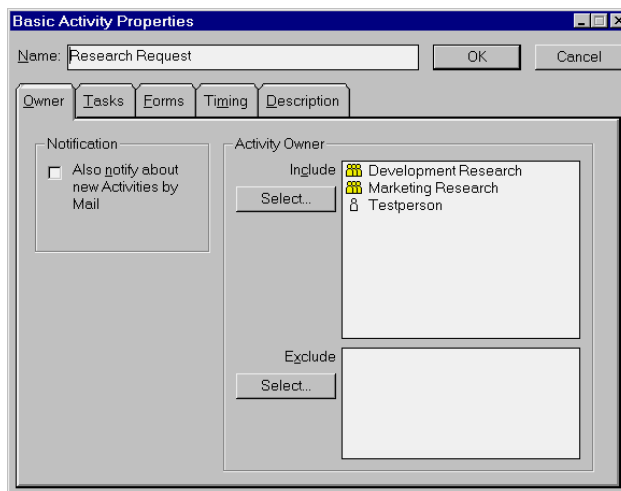
Note Reassign is an advanced property. You need to open the Advanced Properties dialog to access it.



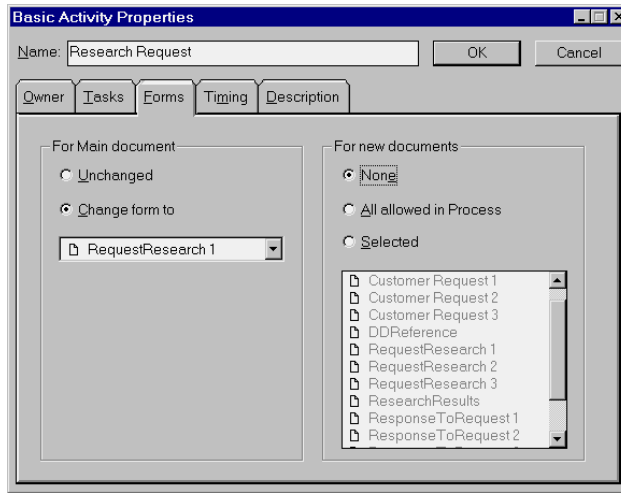
Defining the research request activity

The second activity in the basic scenario is Research Request. In the same way you set the properties for the first activity, do the following:

1. Name the activity **Research Request**.
2. The people who work on the second activity are the researchers, so assign the two research workgroups as the potential activity owners, as in the following figure. Also, don't forget the Testperson role (that's you).

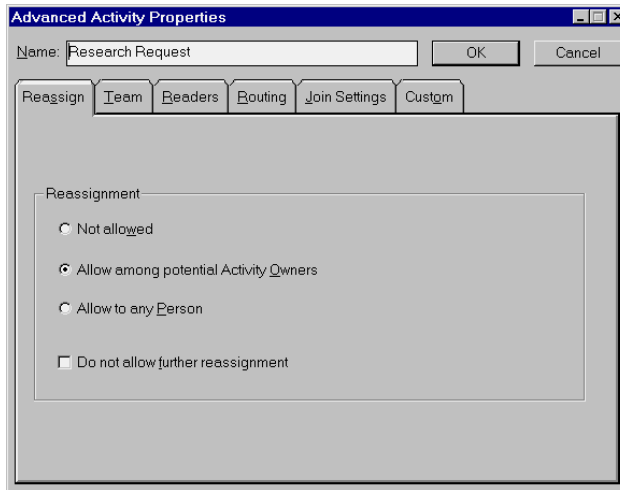


3. On the Tasks tab, enter **Provide Research Information** as the single required task.
4. Researchers use the same main document as the creator of the request. However, they use a different form to see and edit it so select the Change form to button, and select the RequestResearch 1 form from the list box. Then select None for other documents. The Forms property should look like the following figure:



5. Enter a description for the activity on the Description tab.
6. Click OK to close the basic properties dialog box and open the advanced properties settings dialog box for your activity by right-clicking it.

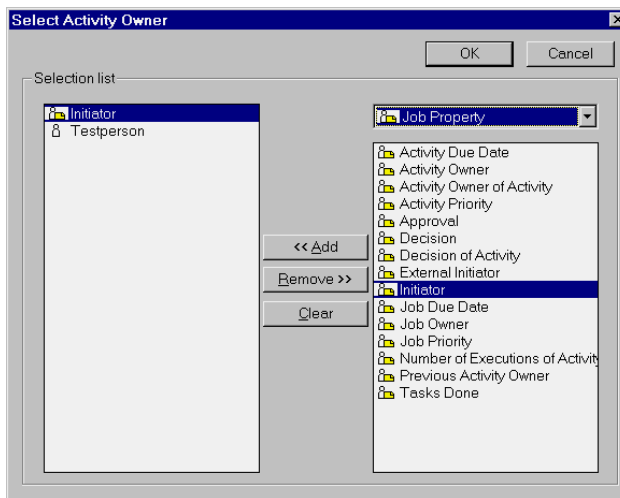
7. Researchers can reassign this activity among themselves but they cannot reassign it to anyone else, so set the Advanced property to Reassign as shown in the next figure:



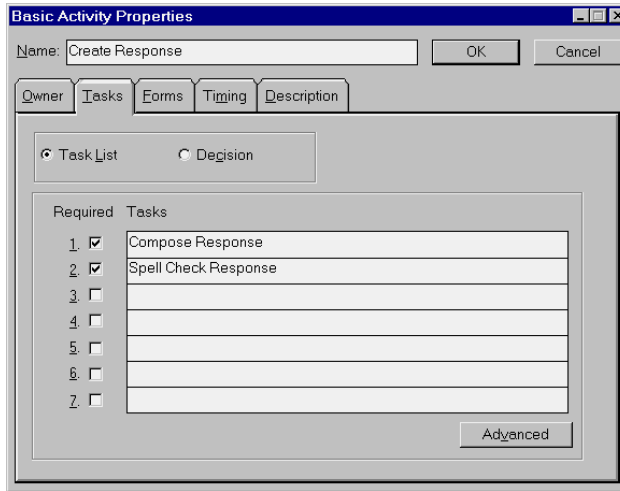
Defining the create response activity

Define the third and last activity as follows:

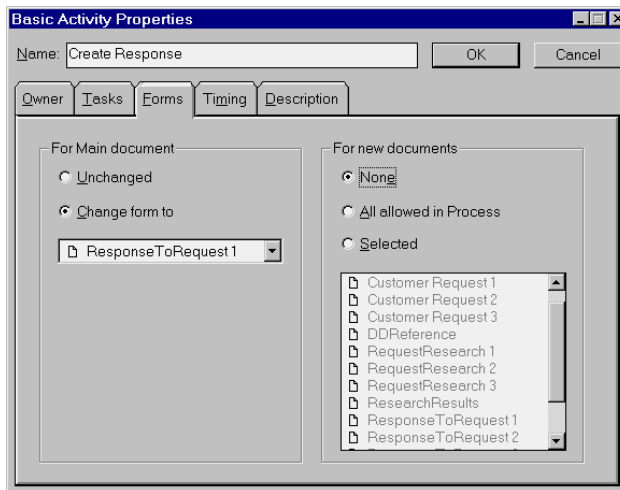
1. Name the activity **Create Response**.
2. We want the person who created the request to also create its response. To enforce that, select the predefined job property, Initiator, for the activity owner, as in the following figure:



3. Enter the tasks for this activity as in the following figure:



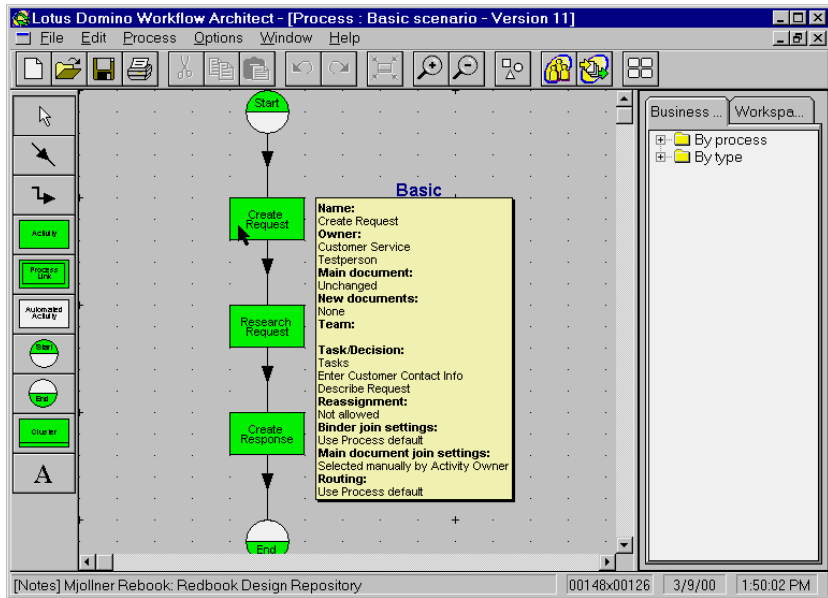
4. When writing a response, the activity owner uses a third form, ResponseToRequest 1, for the main document. Therefore, set the Forms property as shown in the following figure:



5. Enter a description for the activity.
6. Close the basic properties dialog box and open the advanced activity properties dialog box.
7. Since only the initiator of the request is to write the response, set the advanced Reassign property to Not Allowed.

Saving and activating the process

You are almost finished defining your first process. The Architect window should look something like the following figure:



Tip The box in the above figure lists almost all the properties for the Create Request activity. It displays when the mouse “hovers” over a diagram object. You can set the Architect to display object properties by selecting the menu item Options - Preferences, and then choosing your Display Property preferences in the resulting dialog.

All that is left to do now is check the process syntax, and then activate and save it.

Choose the menu item Process - Check Syntax. The Architect will report any syntax errors. Correct any errors found, and check the syntax again.

When the process syntax checks out OK, you can save and activate the process by choosing the menu item File - Activate Process.

Activating a process saves the process definition in the process definition database. End-users can now use the process to create new customer request jobs.

Activating the process also saves the process design in the design repository database.

Note You can save an unfinished process design in the design repository without first activating it.

Congratulations! You have just finished defining your first process. Try it out by opening the application database in the Notes client and starting a new job.

Second prototype: adding more basics

The first scenario's routing relations are very simple, and there are some "flaws" in the workflow design.

For example, all requests get routed to the Research Request activity, where *both* marketing and development researchers are the potential activity owners. Both types of researchers have to look at *each* incoming request to decide whether they need to claim it or not.

We want to improve the workflow so that a request goes only to one group of researchers, so we split the second activity into two research activities, one for each department. We then have the creator of the request decide where to next route the request.

We also want to tailor somewhat how each research group sees the request without having to design separate research forms. We also want to automatically capture in the document which group did the research.

Finally, we want to specify the occasional involvement of other members of the marketing and development departments.

We achieve these aims by using more of the basic Domino Workflow features. The features we use in this prototype are:

- Exclusive Routing
- Team Attribute
- Custom Attributes

Prototyping strategies

Before starting to make your changes to your first prototype, you need to choose a strategy for making the changes. You can use one of two prototyping strategies:

- Directly change the process definition.
- Make changes to a copy of the process definition.

Directly change the process definition

Domino Workflow supports versioning of process definitions. When you activate a later version of a process, the previous version is “frozen”, which means:

- New jobs are started using the latest version.
- Uncompleted jobs continue to be processed with the earlier version, to avoid inconsistencies.

Caution While Domino Workflow supports versioning of process definitions, forms are a different matter. Therefore, you may want to make your changes to a renamed copy of the form.

To make a new version of a process:

1. Make your changes to the process in the Architect.
2. When satisfied with the changes, save and activate the new version.
3. Since there is a previously activated version of the process, you will be asked if you want to “freeze” the old version, and activate the new one. Answer Yes to do so.

Make changes to a copy of the process.

This strategy implies changing the name of the process, since process names must be unique within any one process definition database.

With this strategy, you will need to take special measures to stop users from creating new jobs in the old process:

1. Go to the Administration/Setup view in your application database.
2. Open the Setup document for editing.
3. In the application database’s setup document, expand the Initiation Settings section.
4. Click the “use selected Processes only” check box. Then use the browse button next to that to choose only those processes you want to use for this application database. That is, *don’t* select the old process.

Note This check box controls new job initiation only. Jobs already started under the old process will continue under that process until they finish.

Once all jobs initiated under the old process finish, you should, for performance reasons, delete the old process from the process definition database. To completely delete a process from the process definition database, you must delete *all* documents for *all* versions of the process.

Which strategy to use

Use the “direct change” strategy when making small, incremental changes to a process.

Use the “change copy” strategy when:

- The changes are so substantive and widespread that the resulting process can be considered an entirely new workflow.
- You are developing on your sandbox server, and you want to test, compare and evaluate the different changes you are making, before deciding on the final version.

Because this chapter is a teaching tool, we use the “change copy” strategy, even though each prototype is only slightly changed from the previous one.

Creating forms and process copies for the second prototype

For the second scenario we will be making changes to the application forms. Before creating the second process in the Architect, make copies of the three application forms and change their names as follows:

- Rename Customer Request 1 to Customer Request 2.
- Rename RequestResearch 1 to RequestResearch 2.
- Rename ResponseToRequest 1 to ResponseToRequest 2.

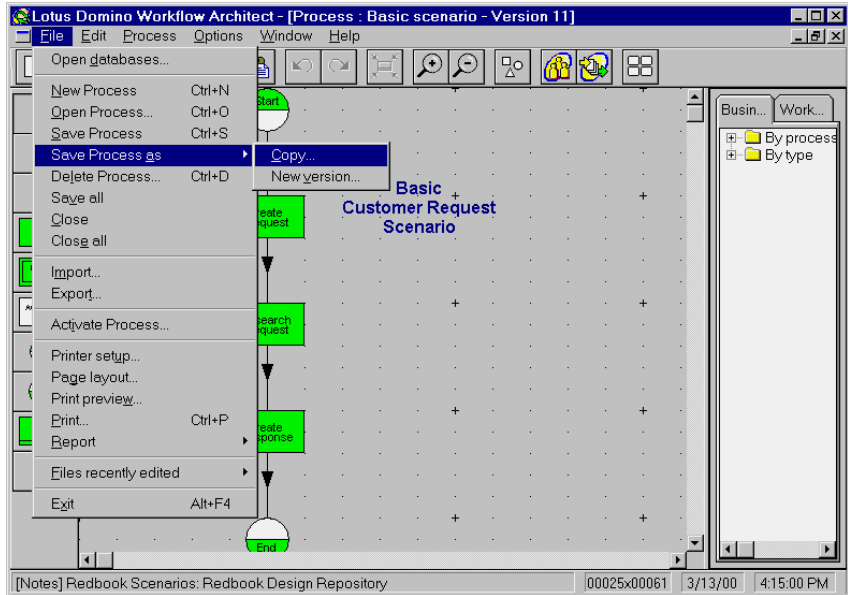
We discuss the changes needed for the research and response forms when we discuss Custom Attributes later in this section.

Note The customer request form does not change in this scenario, so strictly speaking you don’t have to copy it. We copy and rename it just to keep all the form names consistent from prototype to prototype.

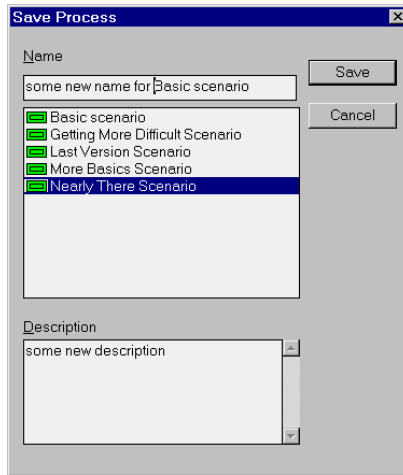
Creating the process from a copy

Now create the second process prototype by doing the following:

1. In the Architect, open the first Basic Scenario process (if it's not already open).
2. Choose File - Save Process as - Copy menu item, as shown in the next figure:



3. Enter a new name and description for the process copy in the Save Process dialog, as in the following figure:



Caution When the dialog box first pops up, the name of the process you are saving appears in the name field and its description in the Description field. As you type in your new name, other processes in the process list will be highlighted. Ignore them.

4. Save the process with its new name and description by clicking Save.

Making changes to the process and activity properties

When you made your copy of the process, all process properties (except name and description) and all activity properties remained the same as in the original. Thus, you don't have as much work to do when defining this prototype as you did with the first.

Changes to process properties

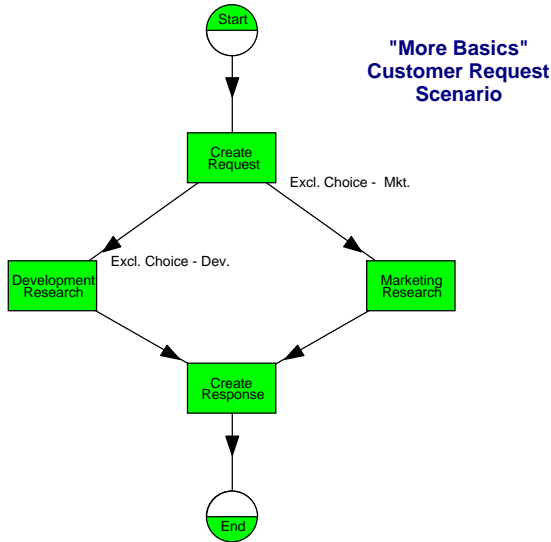
The only other *process* property you change is the Forms property:

1. Access the basic properties for the process.
2. On the Forms tab, select Customer Request 2 for the Main document.

Note The change to the process Forms property is not strictly necessary, since no changes are made to the Customer Request 1 form. However, to keep a consistent form naming convention across the sample prototypes, we copied and renamed the customer request form, and we assigned that new name to the process.

Creation and basic modifications of the research activities

When the second prototype is completed, its diagram will look like the following figure:



To create two research activities and make some basic modifications to their properties, use the following steps:

1. Move the Research Request activity box over to the left side of the diagram.
2. With the activity still selected, use Edit - Copy and Edit - Paste to make a copy of the activity and place it on the right side of the diagram.
3. Connect arrows to and from the copy, as shown in the previous figure.
4. Change the names of these two activities to Development Research and Marketing Research, as shown in the figure.
5. In both research activities modify the Forms property by changing the main document form selection from RequestResearch 1 to RequestResearch 2.

Note The actual form changes are discussed later in this section.

Modifying who participates in the research activities

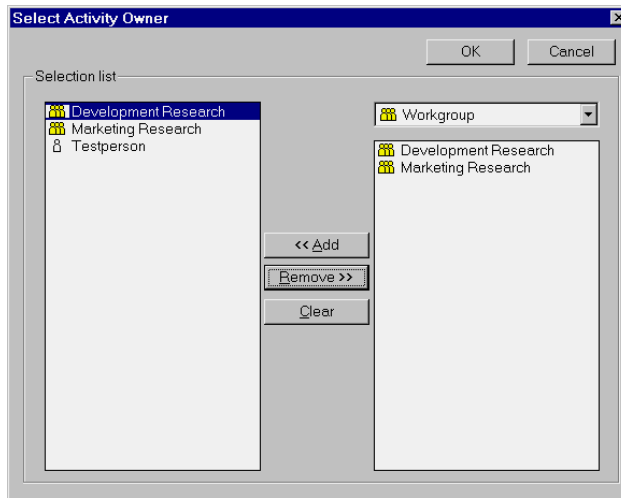
For each of the two research activities, you need to remove the opposite research group as potential activity owners.

For instance, for the Marketing Research activity, you would do as follows:

1. Access the basic properties for the activity.
2. On the Owner tab, click the Include, Select button.

3. In the Select Activity Owner dialog box select, from the left-hand list, the research group to remove as owner. Then click Remove.

The next figure shows the Development Research workgroup being removed from the Marketing Research activity.

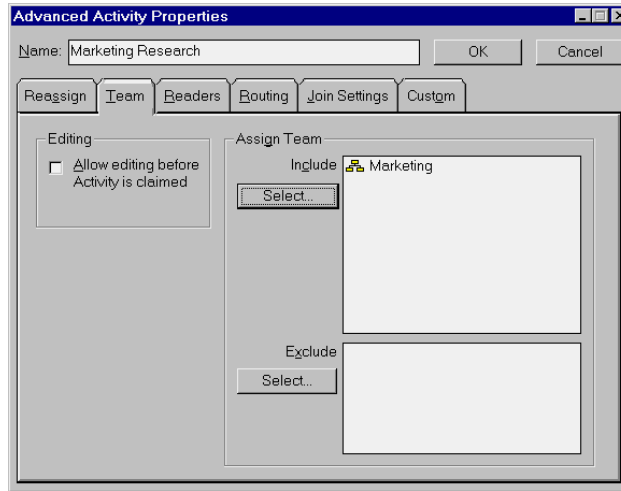


4. Click OK to confirm the removal and to return to the properties dialog box.
5. Click OK to save the Owner property change.

We also want to define a team for each of the research activities. Members of a team can assist in an activity and make changes to binder documents. However, they don't own the activity, and thus can't mark the activity as complete.

In the second prototype, you need to assign the department associated with the activity as the team. For instance, the figure below shows the assignment of the Marketing department as the team for the Marketing Research activity.

Note Team is an advanced activity property.



Modifying routing relations

The arrows that you have been drawing so far are called *Routing Relationships*. They define how a job and its binder are routed from one activity to one or more other activities.

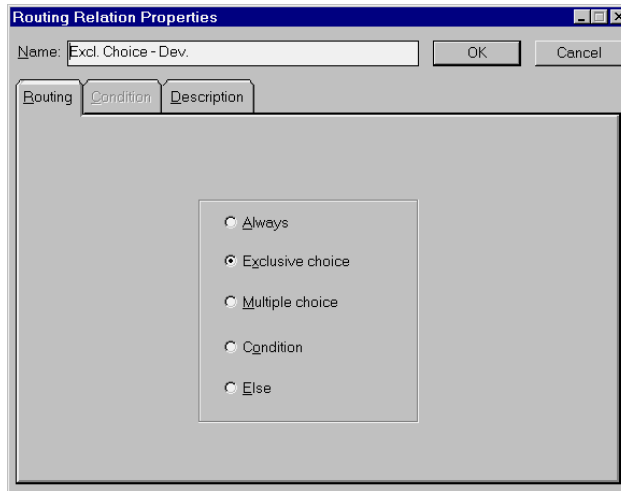
Like activities, routing relations have properties. It is these properties that define how the binder is routed from one activity to one or more other activities. So far, our routing relations (arrows) have used the default properties. Thus each of our arrows indicates that the routing path is *always* taken.

For the second scenario, we want the next activity after Create Request to be *either* Development Research *or* Marketing Research. This is called “Exclusive Choice” routing. To achieve the desired routing, do the following:

1. Select an arrow leading from Create Request and going to one of the research activities. For instance, select the arrow leading towards the Development Research activity.
2. Right-click the mouse, then choose Properties from the pop-up menu.
3. In the Routing Relations Properties dialog box, type in a name for the relation.

Tip The name is used in the routing choice dialog box presented to the end-user when process jobs are run, so enter a name that is descriptive of the choice to be made.

4. On the Routing tab, select the “Exclusive choice” radio button, as shown in the following figure:



5. Click the Description tab, and enter a short description of the routing relationship. Click OK to save the properties changes.
6. Repeat steps 1 through 5 for the other arrow leading from the Create Request activity.

Disabling joins

Our second prototype is an example of multiple parallel routing paths leading from one activity, Create Request, that later join again in the Create Response activity.

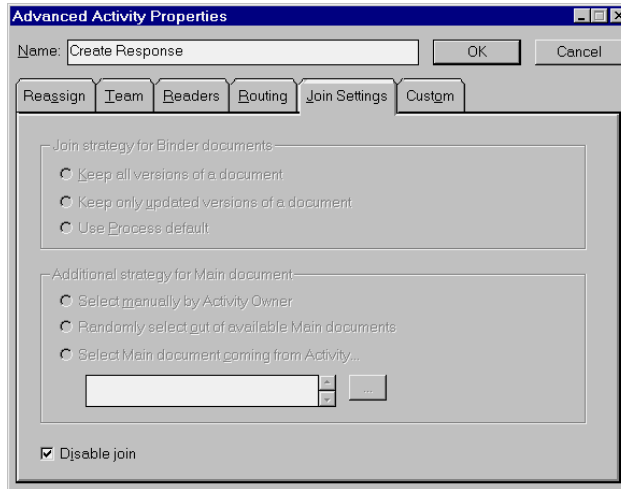
When there are parallel paths, Domino Workflow creates multiple copies of the job binder and its documents. This is to avoid editing conflicts in the different paths.

When the paths rejoin, Domino Workflow then merges all the binder copies back into one binder. This copying and merging is a performance hit. Also, the joining is subject to the scheduling on the server of the application database’s background agent.

Even though our process has multiple parallel paths, any given job going through the process follows only one of those paths. Therefore, we can improve performance by “disabling joins” for exclusive choice routing paths.

For our process, disable the binder copying and merging by doing the following:

1. Select the *Create Response* activity, then access its advanced properties.
Note Set the disable join in the activity where the exclusive choice paths meet up again.
2. On the Join Settings tab, select the “Disable join” check box, as shown in the following figure:



3. Click OK to save the property change.

Custom attributes and how they work

The final changes we want to make in this scenario are:

- Dynamically tailor what each research group sees in the main document, without having to design more than one research form.
- Save a record of the kind of research done for the request in the main document.

We achieve both these aims by using the Custom Attribute feature. Here’s how custom attributes work:

- You define a custom attribute in an activity’s advanced properties.
- When a job enters that activity, Domino Workflow puts that custom attribute, as a Notes item and value, in the main document.
- When the job leaves that activity, Domino Workflow removes the custom attribute (Notes item and value) from the main document.

To make use of custom attributes, you need to make design changes to the form used for the main document in that activity.

Defining the prototype's custom attributes

For our prototype, you need to define a custom attribute in *each* of the research activities by using the following procedure:

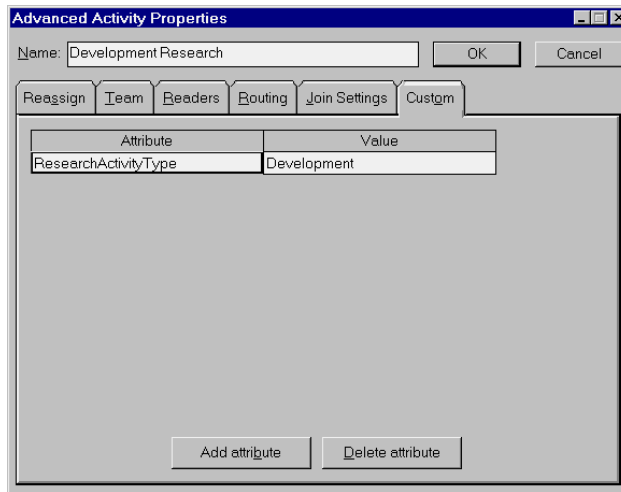
1. Select the activity and access its advanced properties.
2. Click the Custom tab.
3. In the blank space under Attribute type the string **ResearchActivityType**

Note This string is used as the item name in the main document.

4. In the space under Value type the string:
 - **Development** if you are editing the Development Research activity.
 - **Marketing** if you are editing the Marketing Research activity.

Note This string is used as the item value in the main document.

The next figure shows the definition of the custom attribute in the Development Research activity.



5. Click OK to save the custom attribute definition.

Tip Sometimes you can't click OK if focus was last in the Value field. If that happens, put focus back in the Attribute field by clicking there. You can now click OK.

Important Since we are using exclusive routing, there is no problem with using the same custom attribute name in two different activities. However, in other processes, with different kinds of routing paths, it may be safer to use a different name in each activity.

Using the custom attribute in the research form

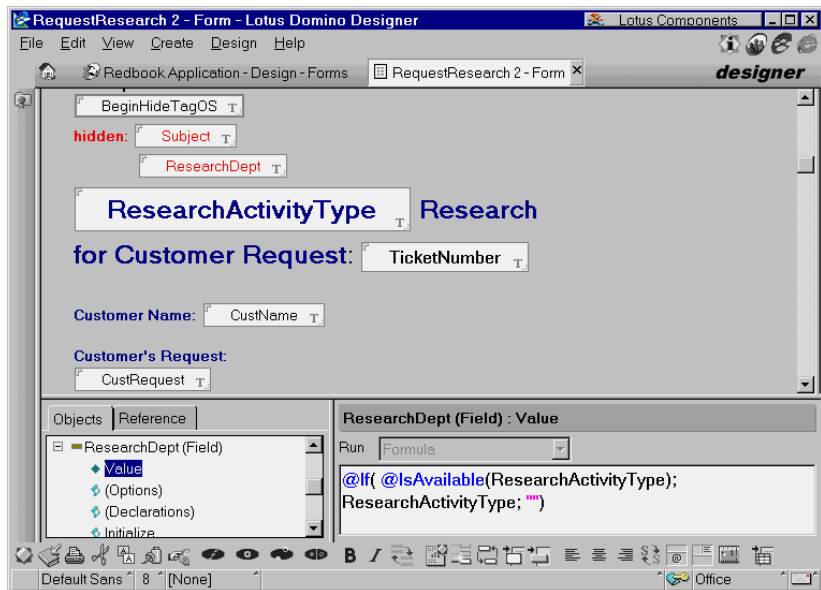
You need to make the following changes to the RequestResearch 2 form:

- Save the value of ResearchActivityType into a field that persists after the activity completes. In our example we put the value of the custom attribute into a field called ResearchDept. It is a hidden field of type Text, Computed with the value:

```
@If( @IsAvailable(ResearchActivityType);  
ResearchActivityType; "" )
```

- Display the value of ResearchActivityType, so researchers see their department name when viewing and editing the research main document.

The following figure shows the design changes we made to the sample RequestResearch 2 form.



Finishing up

You have finished almost all the modifications needed to transform the first prototype into the second.

The only things left to do are:

1. Change the ResponseToRequest 2 form to display the ResearchDept field that was added to the main document during the previous research activity.
2. Change the Create Response activity's Forms property: set the main document form to ResponseToRequest 2.
3. Check the process syntax.
4. If there are syntax errors, make your corrections.
5. When the syntax checks out OK, activate and save the process.

Third prototype: getting more difficult

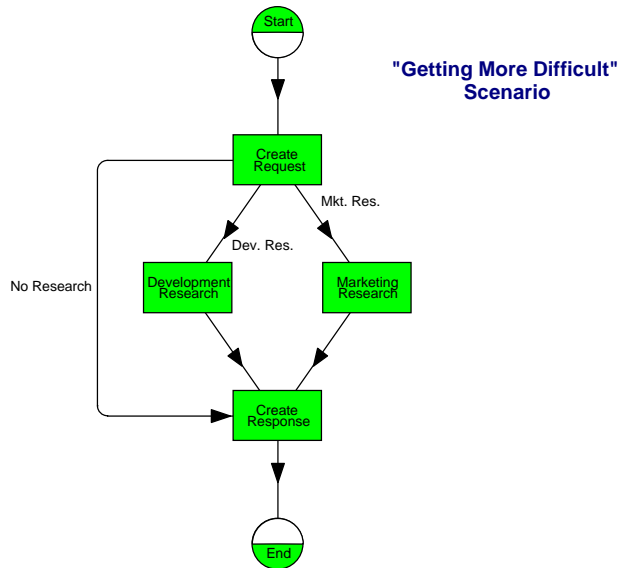
In this prototype, we tinker some more with the way customer request jobs are routed to the research activities. This is done because we observed the following facts about requests:

- Some requests require *no* research at all.
- Some requests require *both* Marketing and Development research.
- Some requests require *either* Marketing *or* Development research.

To implement the above, we use the *Conditional Routing* feature of Domino Workflow. Conditional routing means that a path is taken if a specified condition is true.

In this prototype, the conditions tested for are the existence of particular values in a field of the main document. The initiator of the request will set these values as one of the tasks in the Create Request activity.

Upon completion, the diagram of the third prototype will look like the following figure:



Getting started with the prototype

Use the following steps to create the new process and its main document forms:

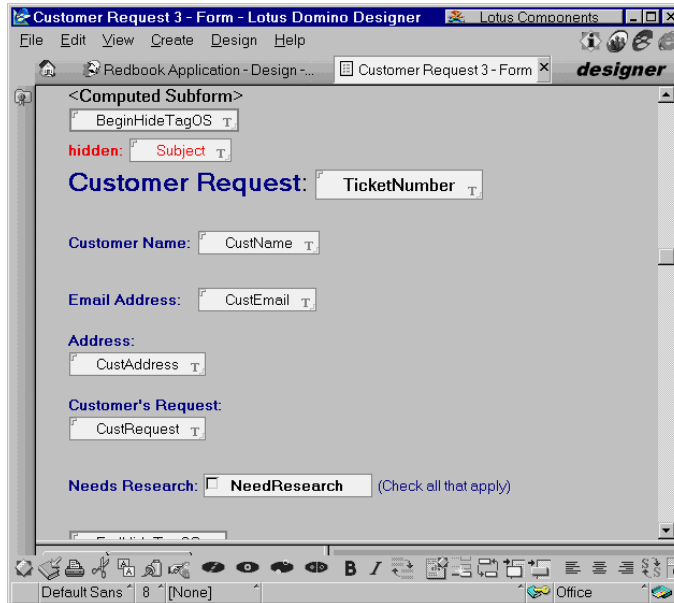
1. In the application database, copy and rename forms as follows:
 - Copy Customer Request 2 and rename it Customer Request 3.
 - Copy RequestResearch 2 and rename it RequestResearch 3.
 - Copy ResponseToRequest 2 and rename it ResponseToRequest 3.
2. Create a copy of the second prototype's process More Basics Scenario. Change the process properties as follows:
 - Name the new process Getting More Difficult Scenario.
 - Enter a new Description for the process. For instance, type in **conditional routing and joining binders**.
 - Set the main document form to Customer Request 3 in the process Forms property.

Modifications to the create request activity

Before you can modify the routing relations leading from the Create Request activity, you will need to do the following:

1. In the Notes Designer, edit the Customer Request 3 form as follows:
 - Add a new field named NeedResearch.
 - Set its type to Checkbox and Editable. Allow multiple values.
 - Set two choices for the check boxes: Marketing and Development.
 - Do *not* allow values not in the list.

The following figure shows how we modified the request form:



2. Change the Tasks property of the Create Request activity as follows:
 - Add a third task.
 - Check new task as required.
 - Enter its description, **Determine if research is needed.**

Changing the routing relations

To effect conditional routing, you need to do the following:

- Change the routing to Development Research to the conditional type.
- Change the routing to Marketing Research to the conditional type.
- Add a new routing leading from Create Request.

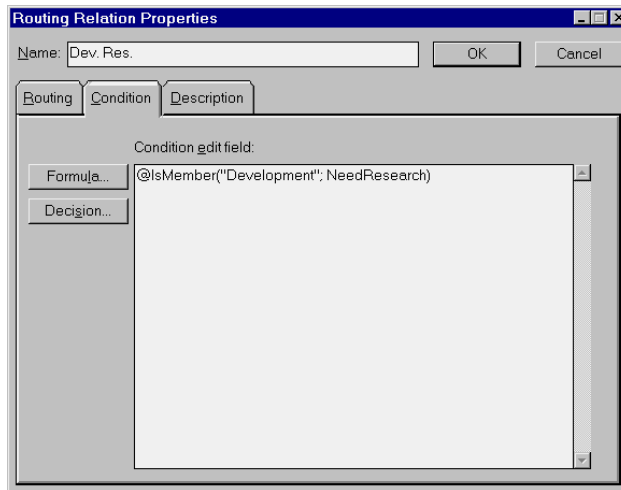
Modifying the routing relation going to Development Research

Make your modifications as follows:

1. Click the arrow leading from the Create Request activity to the Development Research activity.
2. Access its properties, and change its name to **Dev. Res.**
Also change the description to **Needs research from Development Dept.**
3. Click the Routing tab and then select the Condition radio button.
4. Click the Condition tab and enter the following Notes formula in the Condition edit field:

@IsMember("Development"; NeedResearch)

The following figure shows what the Condition tab should look like:



Note Selecting either of the buttons, Formula or Decision, will give you formula editing dialog boxes. These have various selection lists and buttons to help you construct complex formulas. For more information on formulas and how and where to use them, see the Domino Workflow Help database.

Modifying the routing relation going to marketing research

Make your modifications as follows:

1. Click the arrow leading from the Create Request activity to the Marketing Research activity.
2. Access its properties, and change its name to **Mkt. Res.**
Also change the description to **Needs research from Marketing Dept.**

3. Click the Routing tab and then select the Condition radio button.
4. Click the Condition tab and enter the following Notes formula in the Condition edit field:

```
@IsMember("Marketing"; NeedResearch)
```

Adding an “Else” Routing

When using conditional routing, it’s a good idea to provide an “else” path in case none of the conditions are met.

In some scenarios, the else path may lead to an error handling activity. But for this chapter’s third prototype, the else path is an actual requirement of the workflow design.

Define the new routing path as follows:

1. With the angled arrow tool, draw a routing from the Create Request activity to the Create Response Activity.
2. Access its properties, and name the routing relation **No Research**.
3. On the Routing tab, select the Else radio button.
4. Click the Description tab, and enter **Customer Service Dept. has expertise to handle request**.

Enabling other new documents in job binders

In our previous scenario, which used exclusive choice routing, we only had a single main document in the job binder. Now that a request can be routed to *both* research groups, we want to minimize editing conflicts while still having only one main document. We can do this by allowing researchers to add their comments as new additional binder documents, using a special research form.

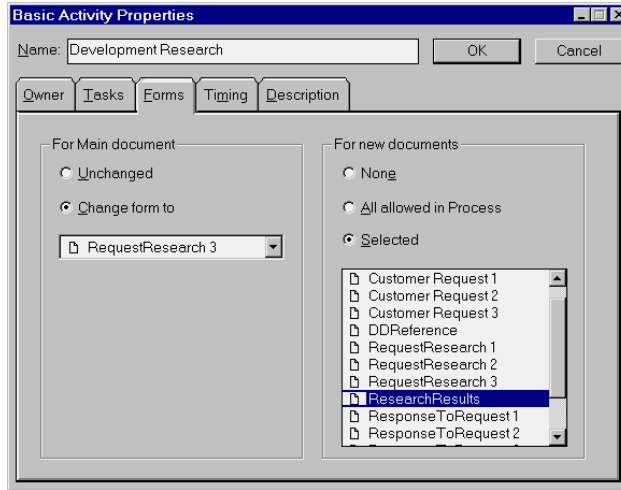
To make the changes for new documents, do the following:

1. Modify the main document form, RequestResearch 3, so that none of its fields are editable. Add instructions telling the researcher to create a new document for the research results. The next figure shows the sample form we used.
2. Create the new document form in the application database as follows:
 - As you did for the first prototype, base your form design on a copy of the Sample form, and maintain the required Domino Workflow fields.
 - Name the new form **ResearchResults**.
 - In order to inherit the custom attributes from the main document, check the form property Formulas inherit values from selected document.

- Add non-editable fields displaying customer request data. Add other editable fields for the researchers to enter their results. The figure below shows the sample form we used.

3. For *each* of the two research activities, Development Research and Marketing Research, change the Forms property as follows:
 - In the “For Main document” section, keep the Change form to button selected but select the RequestResearch 3 form in the list box under that button.
 - In the “For new documents” section on the right side of the tab, select the Selected radio button.
 - In the list box under that button, select the form ResearchResults.

The forms property should now look like the following figure:



Modifying the create response activity

Because a job can now concurrently go through two routing paths, you need to re-enable binder joins at the activity where the paths rejoin. That activity is Create Response.

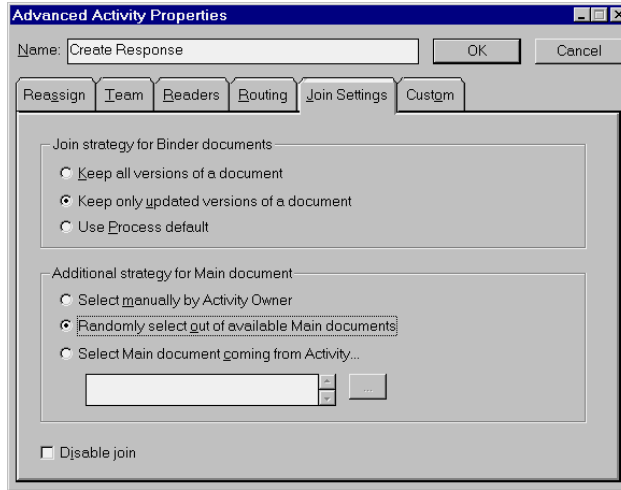
You also need to specify the new main document form name.

Note No changes were made to the second prototype's response form. However, in order to keep a consistent form naming convention across prototypes, we made a copy of the form and renamed it.

Make the following changes to the Create Response activity:

1. Access the basic properties and change the main document form to ResponseToRequest 3.
2. Access the advanced properties and click the Join Settings tab.
3. Deselect the Disable join check box.
4. Since we want to keep the new documents created by researchers, select the Keep only updated versions of documents radio button.
5. Since the main document can't be changed during the research activity, it doesn't matter what main document is used in the join. Therefore, select the Randomly select out of available Main documents radio button.

The Joins Settings tab should look like the following figure:



Finishing up the third prototype

The third prototype definition is now complete. All you need to do is:

- Check the process syntax.
- Make any necessary corrections.
- When the syntax checks out OK, activate and save the process.

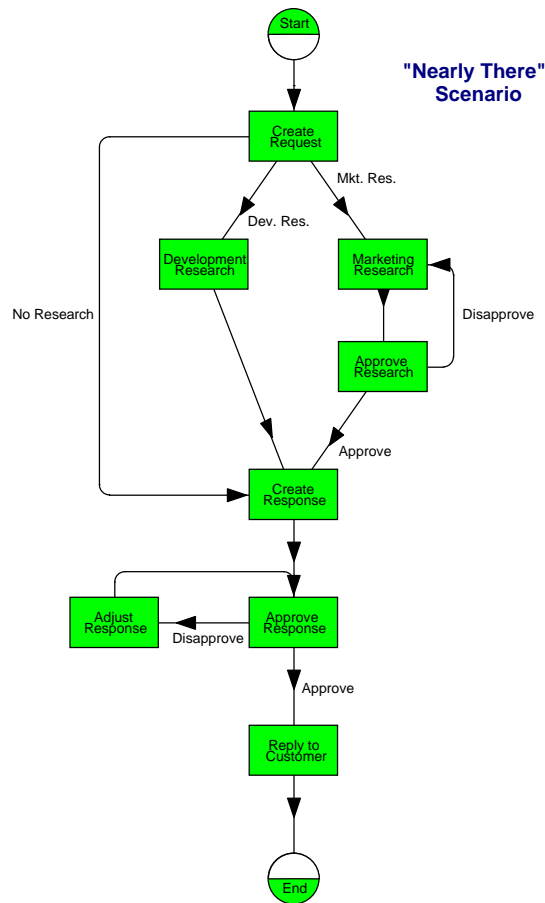
Fourth prototype: nearly there

In this scenario we add:

- A new activity, Reply To Customer
- Two “approval cycles” to the workflow

Note They are called *cycles* because, in the diagram, the not-approved path loops “backwards” in a circular fashion. The job only moves “forward” when approved.

The final process diagram for the fourth prototype looks like the following figure:



As you did in the previous prototypes, create the new prototype by copying the third prototype's process, and rename it Nearly There Scenario.

This scenario uses the same forms as the previous scenario, so you don't have to create new ones.

Defining the marketing research approval cycle

There is a requirement that the results of marketing research be approved before the request is sent on to the Customer Service department. If the research is not approved, then the job goes back to the Marketing Research activity for more research.

Create the approve research activity

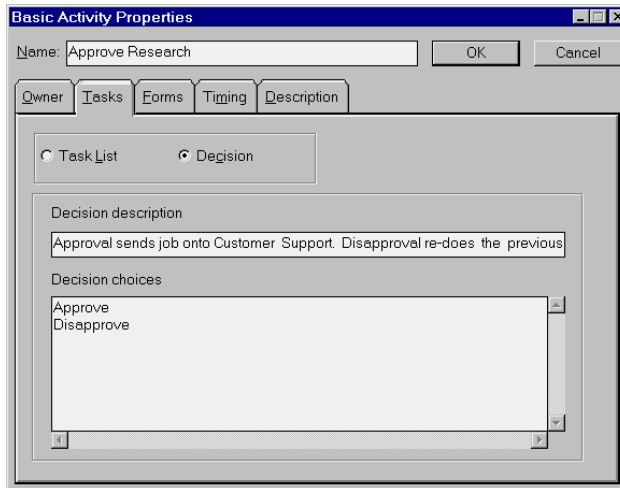
The marketing approval cycle consists of a new activity and two condition routing paths. To create the new activity, do the following:

1. Create a new activity between Marketing Research and Create Response.
2. Set its basic properties:
 - Name the activity **Approve Research**
 - Include the Marketing Research workgroup and the Testperson role as the activity owner
 - Enter the description **Approve research results**
 - For Forms property, Select the Unchanged button for the Main document, and the None button for new documents.
3. No tasks are performed in this activity. However, an approval decision is made during this activity so on the Tasks property tab, click Decision.

Now enter a description of the decision to be made. This description is placed in the job's cover document, so it should tell the activity what to do.

Enter two decision choices, **Approve** and **Disapprove**. These two decision choices are made available as job properties. The routing paths leaving this activity can then use these values in condition formulas.

The Tasks tab should now look like the following figure:



4. Set the advanced Reassign property to Allow among potential Activity Owners.

Create the disapprove routing path

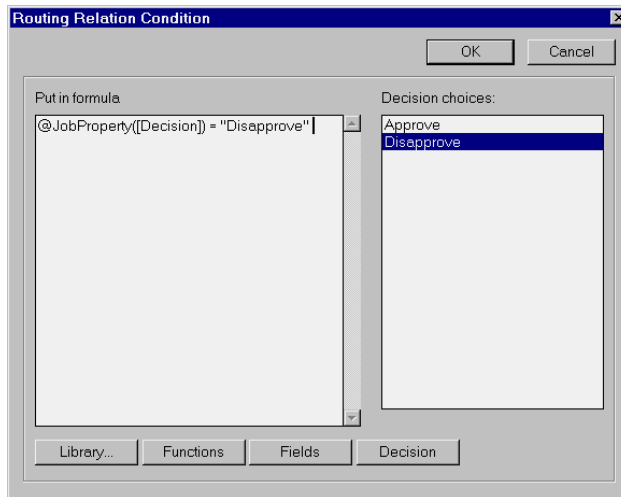
From the new Approve Research activity, you need to create a new path (draw an arrow) back to the Marketing Research activity. Then set its properties as follows:

1. Set its name to Disapprove and its description to **sends job back to previous activity for more research.**
2. On the Routing tab, click the Condition radio button.
3. On the Condition tab, click Decision, to bring up the edit decision-formula dialog box.

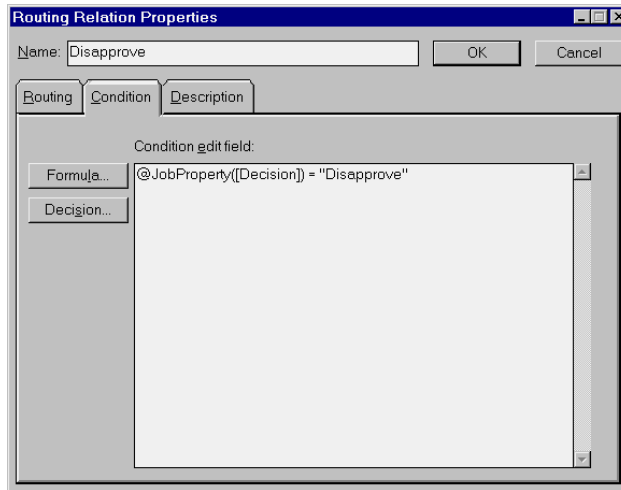
Double-click Disapprove that appears in the Decision choices list on the right-hand side of the dialog box. That action puts the following job property formula in the left-hand formula box:

```
@JobProperty([Decision]) = "Disapprove"
```

The formula dialog should look like the following figure:



4. Click OK in the dialog box. That brings you back to the properties dialog box. The Condition tab should now look like the following figure:



5. Close the properties dialog box and save your settings by clicking OK.

Create the approve routing path

To create the approval path, you need to modify the existing routing relation (arrow) leading from the Approve Research activity and going to the Create response activity.

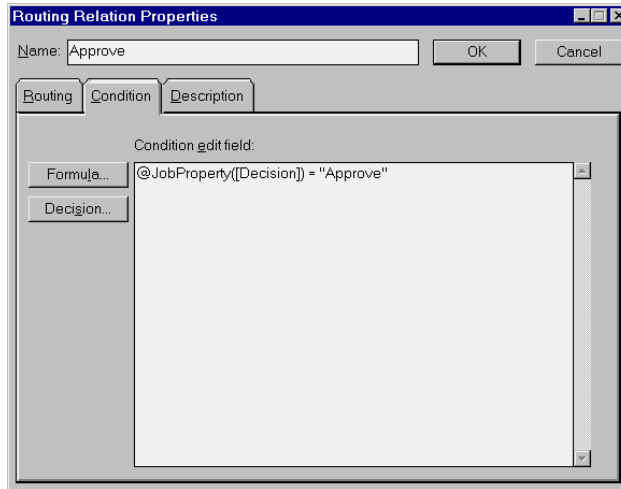
The changes you make are similar to the ones you made previously for the disapproval path. They are:

- Name the routing relation Approve and put in the description **Approval sends job on to Customer Service.**
- Set routing type to Condition.
- Using the same steps as you did in the Disapprove path, except that you choose Approve, define the condition formula:

```
@JobProperty([Decision]) = "Approve"
```

Tip Now that you know how to write the decision formula, you can choose to type it in directly instead of using the formula-writing dialog boxes.

The Condition tab should now look like the following figure:



And that's it for creating the marketing research approval cycle.

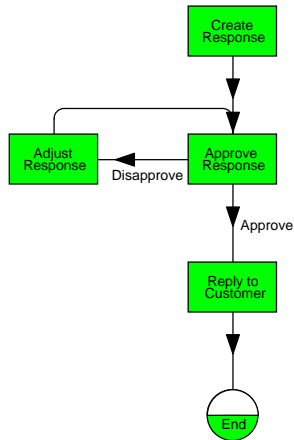
Defining the response approval cycle

The response approval cycle is similar to the marketing research cycle you just defined, and you define it in a similar fashion.

The "extra" feature here is that instead of looping the Create Response activity, the disapprove path goes to a newly defined Adjust Response activity. And to get the *cycle*, the Adjust Response activity loops back to the Approve Response activity.

Note We introduce the Adjust Response activity so that we don't loop back to an activity where parallel paths are being joined.

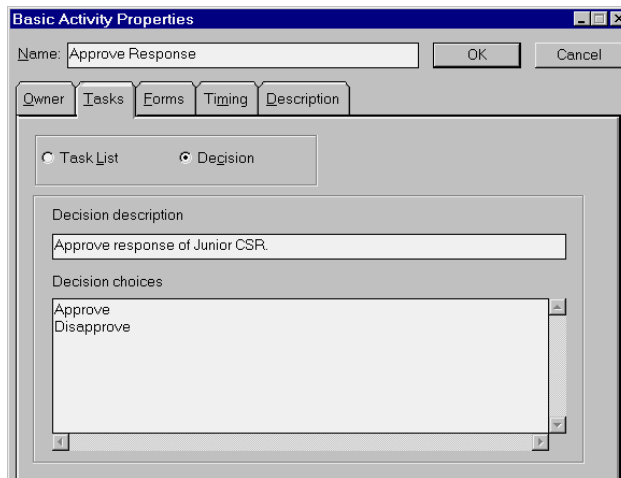
Draw activity boxes and arrows between Create Response and the End node to match the diagram excerpt shown in the following figure:



Define approve response properties

Set the properties of the new Approve Response activity as follows:

- Name it Approve Response and type the description **If a Junior CSR wrote the response, then a CS Supervisor must approve it.**
- Select the roles CS Supervisor and Testperson as the activity owners.
- Set the Tasks property as shown in the following figure:



- Set the advanced property Reassign to Allow among potential Activity Owners.

Define approval and disapproval routing relations

Just as you did for the marketing approval cycle, set the routing relations (arrows) leading out of the Approve Response.

- The arrow going to Reply to Customer activity is the Approve path. Use the same techniques you used for marketing cycle to define its properties.
- The arrow going to the Adjust Response activity is the Disapprove path. Again use the same techniques you used in the marketing cycle to define its properties.

Define the adjust response activity

This activity is very similar to the Create Response activity. The “extra” here is that we use a formula to assign the activity owner.

The formula is evaluated when the job is run, and it specifies that the person who was the Create Response activity owner is to be the potential activity owner of Adjust Response activity.

Do the following to define the activity’s properties as follows:

Note We show how to define the activity owner formula in the next subsection.

1. Name the activity Adjust Response and type the description **The response needs adjustment or correction.**
2. Set the Tasks property as shown in the following figure:

The screenshot shows the 'Basic Activity Properties' dialog box for the activity 'Adjust Response'. The 'Name' field is set to 'Adjust Response'. The 'Tasks' tab is selected, showing a list of tasks under the 'Required' column. The first two tasks are checked: '1. Re-write the response' and '2. Spell check response'. There are also buttons for 'OK', 'Cancel', and 'Advanced'.

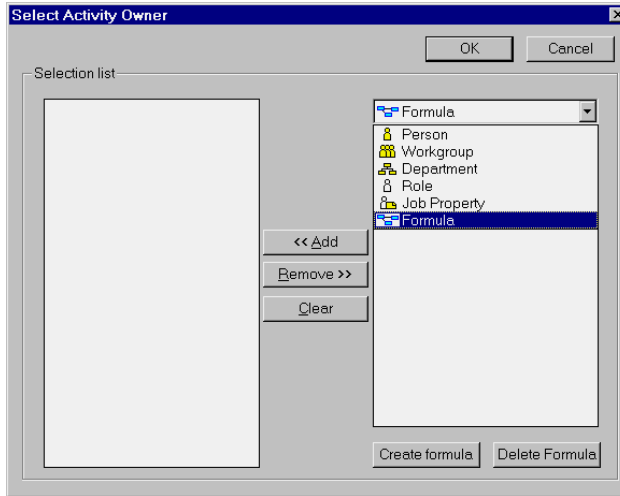
Required	Tasks
<input checked="" type="checkbox"/>	1. Re-write the response
<input checked="" type="checkbox"/>	2. Spell check response
<input type="checkbox"/>	3.
<input type="checkbox"/>	4.
<input type="checkbox"/>	5.
<input type="checkbox"/>	6.
<input type="checkbox"/>	7.

3. Set the Main document form to ResponseToRequest 3.
4. Set the advanced Reassign property to Not Allowed.

Defining the owner formula

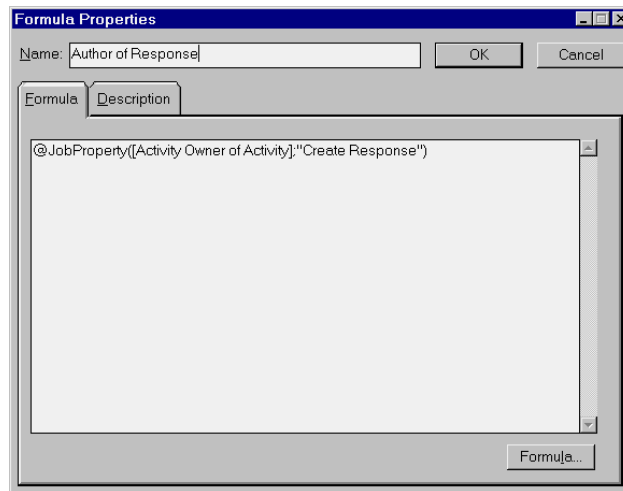
Just getting to the point where you can actually define the formula is a bit torturous. You have to go down through several dialog levels, as follows:

1. On the Owner property tab, click Include Select to open the Select Activity Owner dialog box.
2. Before you can select the formula, you have to define it. To define the formula, select Formula from the right drop-down list box, as shown in the following figure:



3. Click Create formula to open the Formula Properties dialog box. Set the name of the formula to Author of Response. On the Formula tab enter this formula:
`@JobProperty([Activity Owner of Activity];"Create Response")`

The Formula Properties dialog box should look like the following figure:



4. Click OK to save the formula definition and to close the dialog box.
Now, back in the Select Activity Owners dialog box, add the formula you just created, Author of Response, to the list of owners.
5. Click OK to close the dialog box and save your owner formula.

Defining the reply to customer activity

Defining this activity is straightforward and simple. You set its properties as follows:

1. Name the activity Reply to Customer.
2. Set its description to **send the response to the customer**.
3. Define a single required task, Send the response to the customer.
4. Select the Initiator job property and the Testperson role as the activity owners.
5. Set the advanced Reassign property to Not Allowed.

Finishing up

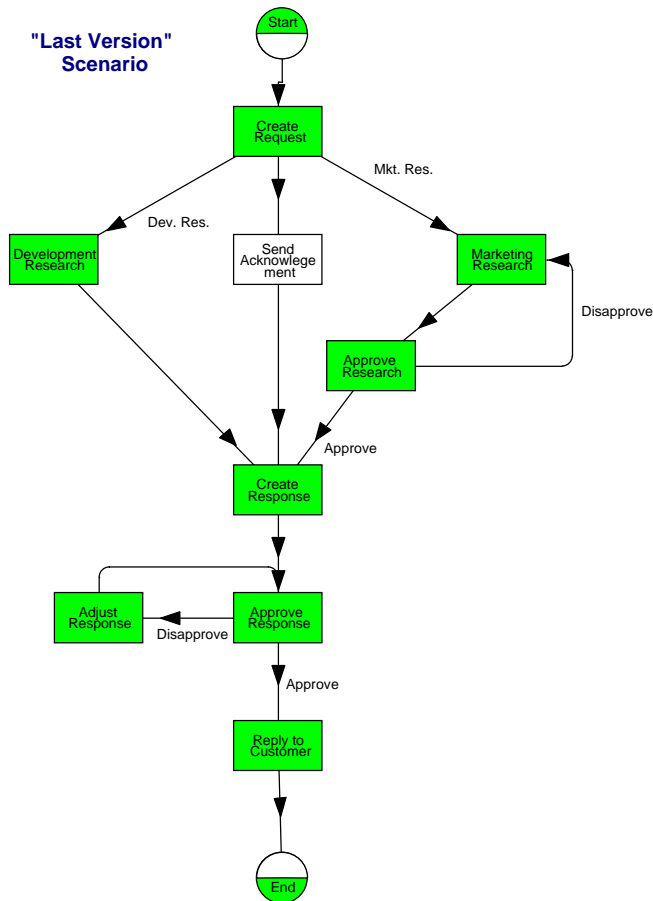
The fourth prototype is now complete. As usual, check process syntax. When it checks out OK, activate and save the process.

Fifth prototype: last version

We want to send each customer an acknowledgment that his or her request is being processed. We also want to include the request's ticket number in the acknowledgment.

We can use an Automated Send Mail activity for sending the acknowledgment, thus lessening the work that the Customer Service Representative has to do when initiating a new request job.

The process diagram for this fifth and final prototype is shown in the following figure:



Follow the usual steps in creating a new prototype:

- Make a copy of the Nearly There Scenario process.
- Rename it Last Version Scenario.

- Give it a new description. For example:
The final version of Customer Request Process. Added automated activity to send the customer a mail acknowledgment.

Defining the automated send acknowledgment activity

The following subsections describe the steps you perform to draw the automatic activity and to set its properties.

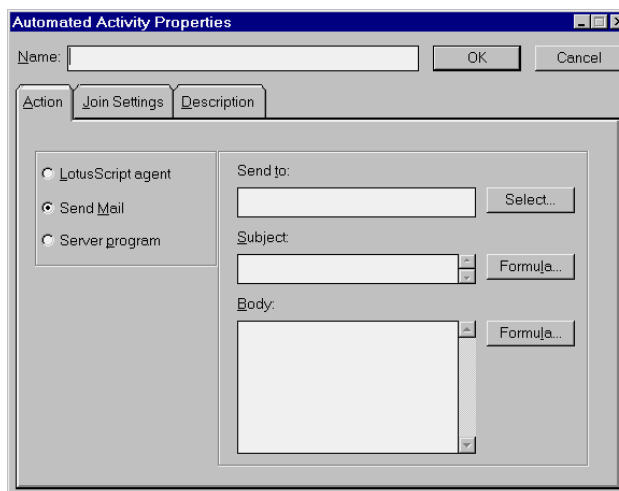
A major part of setting properties for automated send mail activities involves writing formulas that specify values for the mail's Send to, Subject, and Body items. We discuss those properties in a separate subsection.

Drawing the automated activity and setting some of its properties

Create the Send Acknowledgment activity as follows:

1. Draw the activity and its routing relations:
 - Using the automated activity drawing tool (the white box), place the activity in the diagram between Create Request and Create Response.
 - Place an arrow going from Create Request to the new activity.
 - Place another arrow going from the new activity to Create Response.
2. Double-click on the automated activity to open the properties dialog box. The dialog box will look like the following figure.

Note Automated activities have only a single set of properties.



3. Enter the name for the activity, Send Acknowledgment.
4. On the Description tab, enter a description for the activity. For our sample, we used:
Send customer the ticket number of the request.
5. The default action for automated activities is Send Mail, which is what we want for this prototype. Therefore, you do not have to set an action type.

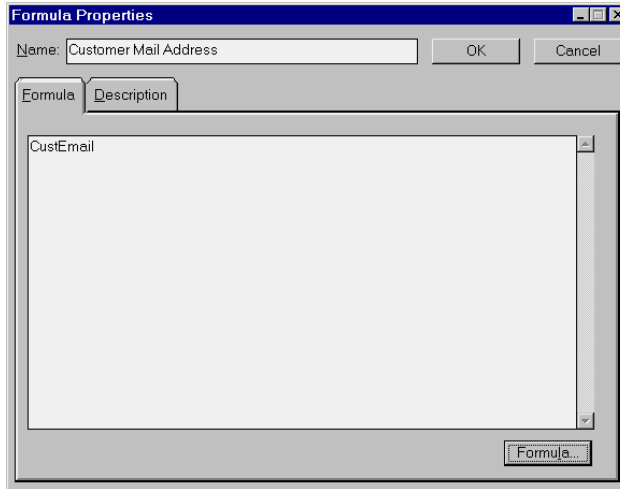
Defining the send to formula

Important The send mail formulas you write will be evaluated when the process runs. They are Notes formulas, and are evaluated in the context of the job's main document.

To specify the Send To formula, do the following:

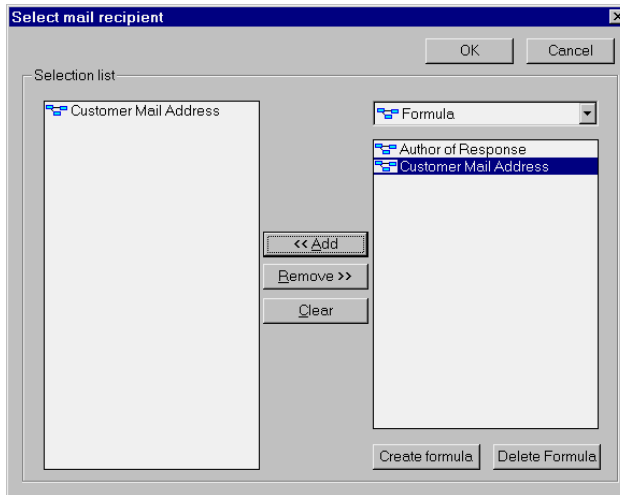
1. Click the Action tab in the Send Acknowledgment properties dialog box.
2. Click Select to the right of the Send to box.
A dialog box similar to the Select Activity Owners dialog box appears. To open a Formula Properties dialog box, select Formula from the drop-down list box.
Click Create Formula.
3. In the Formula Properties dialog box, enter a name for the formula. Our sample uses Customer Mail Address as the formula name.
4. Write the formula to evaluate the Send To value in the edit box on the Formula tab.
For our prototype we want the acknowledgment to go to the customer's e-mail address. That address is stored in the CustEmail field of the main document.

Enter that field name in the edit box. The Formula Properties dialog box should now look like the following figure:



5. Click OK to save the formula definition and to close the Formula Properties dialog box.
6. You are back in the Select mail recipient dialog box. Select the formula you just created, Customer Mail Address.

Click the Add button to add the formula to the selection list. The dialog box should look like the figure below:



Note In this prototype we only send the acknowledgment to the external customer. However, it is possible to send automated mail to others as well. To do that, add recipients to the list in the same manner that you used to choose activity owners.

7. Click OK to save the recipient list.

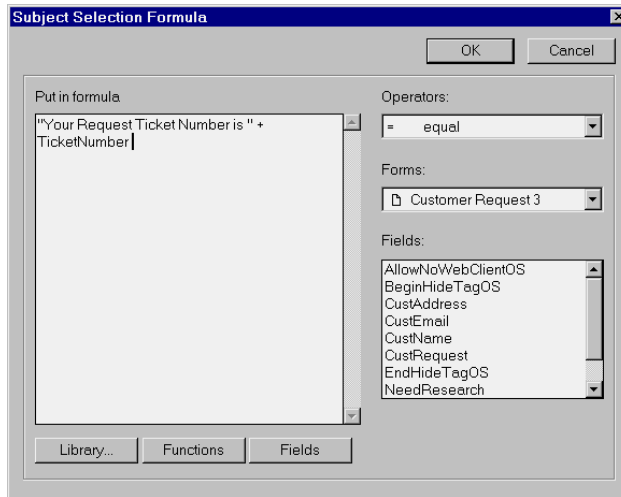
Defining the subject formula

To specify the Subject formula, do the following:

1. Click the Action tab in the Send Acknowledgment properties dialog box.
2. Click the Formula button to the right of the Subject to open the Subject Selection Formula dialog box.
3. Compose a formula in the left-hand edit box. For our sample we used the following formula that puts the request's ticket number in the subject:

"Your Request Ticket Number is " + TicketNumber

The dialog box should now look like the following figure.



Tip Use the various buttons and list boxes to compose more complex formulas, such as those involving job properties.

4. Click OK to save your subject formula.

Defining the body formula

To specify the Body formula, do the following:

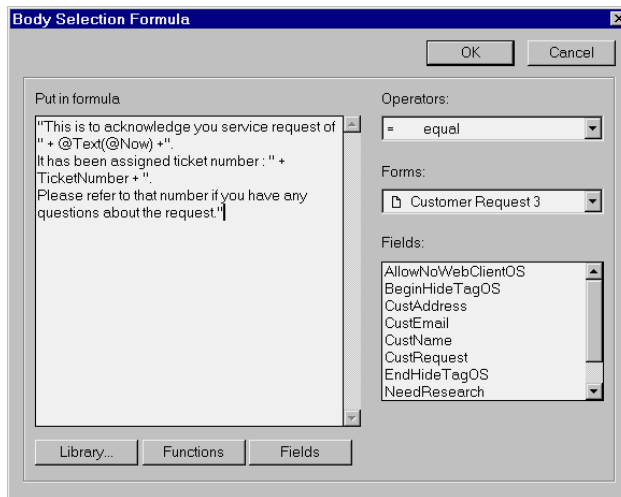
1. Click the Action tab in the Send Acknowledgment properties dialog box.
2. Click the Formula button to the right of Body to open the Body Selection Formula dialog box.
3. Compose a formula in the left-hand edit box. For our sample we used the following formula that puts the request's ticket number in the subject:

```
"This is to acknowledge your service request of " +  
@Text(@Now) +".
```

```
It has been assigned ticket number : " + TicketNumber + ".
```

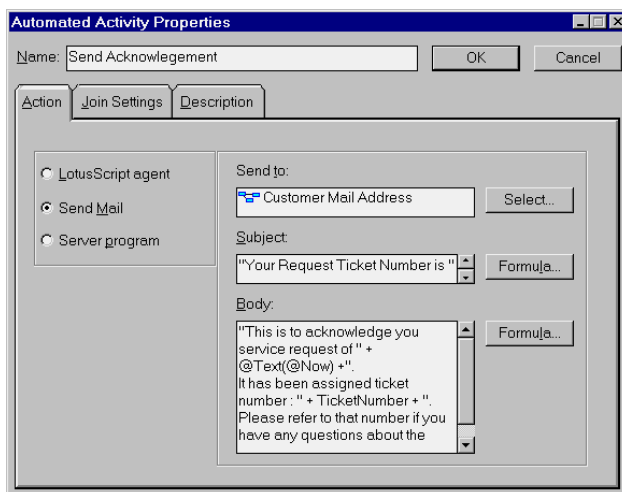
```
Please refer to that number if you have any questions  
about the request."
```

The dialog box should now look like the following figure:



4. Click OK to save your body formula.

After you have finished defining all your formulas, the Action properties tab for the Send Acknowledgment should look like the figure below:



Click OK in the activity properties box to save all your formula definitions.

Adjusting routing and binder joins

If at this point you were to check process syntax, you would get a warning that the Else routing path may not be reachable.

Because the routing to Send Acknowledgment is of type Always, we no longer need the fallback Else path. Therefore, delete the Else arrow.

Due to the always-type routing, we can make a better main document selection choice when binders are merged in the Create Response activity. To make this adjustment, do the following:

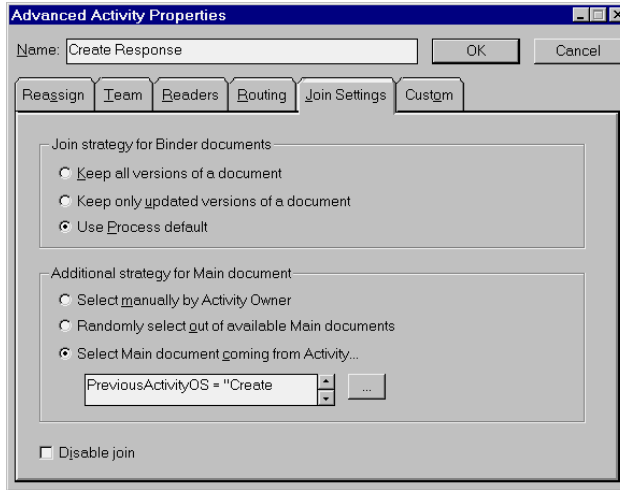
1. Open the advanced properties for the Create Response activity.
2. On Join settings tab, select the button labeled Select Main document coming from activity.
3. We always want to use the main document as it comes from the Create Request activity, since we know it's not edited during the research and send mail activities; and because we now have an always path to Create Response.

You can't use the browse button to make your activity selection since that button presents only the activities that are the immediate predecessors to Create Response.

You need to enter the following in the edit box:

```
PreviousActivityOS = "Create Request"
```

4. The Join Settings tab should now look like the following figure:



Click OK to save your join modifications and to close the dialog box.

Finishing up

The fifth and last prototype is now complete. All you need to do now is check its syntax, make any corrections, and then activate and save it.

Summary

In this chapter we described how to define your first complete Domino Workflow process, using the normal “out of the box” features of Domino Workflow.

We showed you how to define increasingly complex processes by building on an earlier, simpler version.

The following lists, by prototype, the Domino Workflow features covered in this chapter:

1. **Basics Scenario**
 - Use of multiple forms in different activities
 - Basic routing
2. **More Basics Scenario**
 - Use of exclusive choice / multiple routing paths
 - Disable join
 - Custom Attribute

3. Getting More Difficult Scenario

- Conditional routing based on a keyword field
- Joining binders

4. Nearly There Scenario

- Introduced looping (Approval cycle): loop back to previous activity
- Introduced another approval cycle: loop back to a separate activity for adjusting response
- Use of job property “Activity Owner of Previous Activity”

5. Last Version Scenario

- Automated Send Mail activity

Within the examples we also discussed the following Domino Workflow concepts:

- Workgroups
- Roles
- Job properties
- Team activities
- Shared activities

Chapter 5

Accessing Domino Workflow from Web clients

If you want to make your application available to Web clients, you can use the standard Domino features to do this. Domino Workflow is plain Domino, so everything you can do in Domino, you can do in Domino Workflow. Additionally, Domino Workflow offers an out-of-the-box Web user interface that you can use to make your application available for the Web clients.

This chapter describes how you can make the process that was created as the final example in Chapter 4 accessible for participants using Web clients. It happens by means of using the standard Web user interface of Domino Workflow. We also discuss the advantages and disadvantages of using Domino Workflow through a Web client. Finally, we go into some customization issues regarding the Domino Workflow Web application.

How to make your workflow database available to Web clients

There are several scenarios you can consider for Web-enabling your workflow application. First, you might think of allowing Web users to initiate new workflows, but you can also consider offering all the workflow functionality to Web clients.

In the first scenario, only initiation from the Web, you can use the feature mail-based or form-based initiation of Domino Workflow. By using mail- or form-based initiation, you can create a document or send a mail to your application whenever a Web user fills out an order form or files a complaint. The workflow will be initiated in the background automatically. The details of this feature are described in Chapter 7.

In the second scenario, Web-enabling all functionality in your workflow application, you need to do more. To make Domino Workflow work through the Web there are four main things you have to do:

1. Enable the application forms used in the process for Web use.
2. Activate the Domino Workflow Web agents.
3. Adjust the setup document.
4. Update your process cache.

Use of the Domino HTTP server

Before enabling the Domino Workflow application, your environment should be Web-enabled. This means that the Domino server has to run an HTTP task and workflow participants must be able to access the Domino server with their Web browsers. For details refer to the Domino Administration Help.

Enable your application forms

When you enable your application for Web use, you must also Web-enable the forms used in your processes. Domino Workflow provides a standard subform with all necessary HTML code for the workflow actions.

You need to consider whether you want to allow use of your application only by Web clients or by both Web and Notes clients.

In the first case, using only Web clients, you need to replace the "(OS Domino Workflow Information)" subform with the "(OS Domino Workflow Web)" subform on your workflow form.

In the second case, using both the Notes client and Web browser, you should include a computed subform based on a formula that determines what client type you are using. To determine this, you can use the following formula:

```
@If(@ClientType = "Notes";  
      "(OS Domino Workflow Information);"  
      "(OS Domino Workflow WEB)")
```

When you use your forms on the Web, you should be aware of some limitations of Domino. For instance, some @Functions will not work on the Web, like @MailSend, @PickList and @DialogBox. For more information about these unsupported features for Web clients, look in the Domino R5 Designer Help database.

Note On the sample form used in the scenario there is a field called "AllowNoWebClientOS". To enable Web clients, delete this field from your form or give it the value "no". If this field has the value "yes", Web clients will not be allowed to edit.

Activate the Web agents

The workflow activities done by a Web client are transferred to your application by agents. Therefore, you must enable all agents used on the Web with the administrator ID of the server. We recommend using a special Domino Workflow Administrator ID which is allowed to run unrestricted agents on the server.

You must enable the following agents:

- OS Web Activity Completed

This agent is used when the Web user has completed an activity.

- OS Web Claim Binder

This agent is used when the Web user claims a certain binder.

- OS Web New Job

This agent is used when the Web user wants to start a new job. When a new job is started the only command necessary is:

```
@Command([Compose];"(OS Web New Job)")
```

The "OS Web New Job" agent is used in the QueryOpen event of that particular document.

- OS Web NoEditRights

This agent is used when the Web user does not have edit rights to the specific document. An error will be generated.

- OS Web Process Init

This agent is used when a Web user initiates a new job. This is done after giving the process name and job name on the "(OS Web New Job)" form. This is the QuerySave agent of the form "(OS Web New Job)."

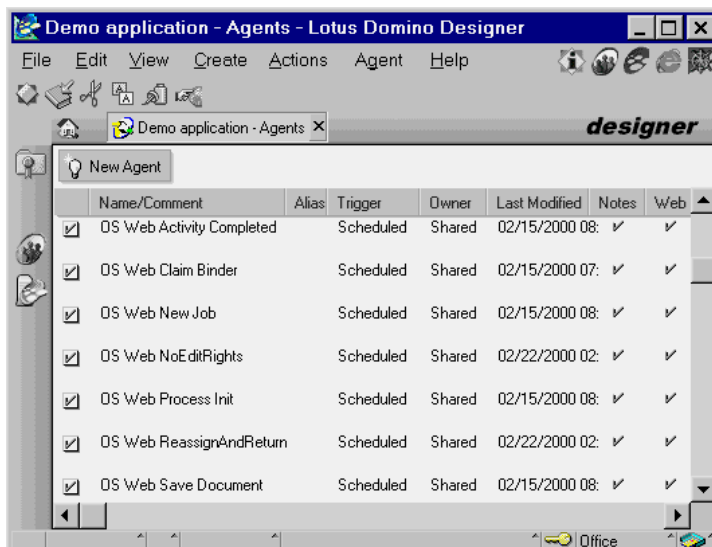
- OS Web ReassignAndReturn

This agent is used when a Web user returns a document after a reassignment. If a document was reassigned using the option "Return Activity after completion," an activity cannot be completed (by the user it was reassigned to), but only returned to the person who reassigned the activity originally.

- OS Web Save Document

This agent is used when a Web user saves a document from the binder.

The signed and enabled agents are shown in the following figure:



You enable the agents the same way as described in Chapter 3 in the section "Enabling and signing Domino Workflow Engine agents."

Note The only reason why all Web agents of Domino Workflow have the setting "on schedule/never" is to require users to enable them before being used. By enabling, the agent is signed by the user. This is important to ensure sufficient rights for these agents.

Adjust the setup document

The setup document in your application database must be changed. Enter a reference to the server name or IP address and enable the process cache (process cache is explained later in this section).

Use the following steps to do this:

1. Open your application database.
2. Choose View - Administration - Application Setup.
3. Open the setup document in this view.

Note If there is no setup document available, the view will be empty. In Chapter 3 you will find information on how to set up the database, including the setup document.

4. Put the document in Edit mode by clicking the Edit Document button.

5. In the section “database settings” fill in the server name or IP address and the path and file name for the application database. In our example we use the server name: myserver.lotus.com and the path and file name: myapplication.nsf

Field	Value
Resource name	
Server's IP address	MyServer.lotus.com
Path	MyApplication.nsf
URL	http://MyServer.lotus.com/MyApplication.nsf/OS+WD+A+Name?OpenView

In the field for server IP address you can either specify an IP address, like 9.95.34.10, or you can give the DNS name of the server, as we did. In most cases it is easier for end-users to recognize the DNS name of the server.

In the field for database path you can either specify the physical path and filename of the database (relative to the data directory) or you can specify the replica ID of the database.

Note If you specify the path, avoid spaces in both the path and the filename, as the resulting URL may not be usable. If you specify replica ID in the path field, the colon (:) has to be removed from the replica ID.

6. In the section initiation settings, you must enable the process cache.
The process cache stores a list of processes that can be initiated in the application. The start time for new jobs can be reduced by using the process cache. Use of the process cache is required when using Web clients. The process cache ensures that processes will appear in the process selection box, when initiating a new job from the Web. Without the process cache you will not be able to start any job from your Web clients.
7. Once it is enabled, you can schedule updates of the process cache by clicking the Schedule button. Schedule the routine based on how often your process changes.
8. Save and close the document.

Tip The URL generated by the setup document can be copied to the clipboard to have a reference to the Domino Workflow application. The setup document has to be in Reader mode to do this.

Update the process cache

The process cache will be updated on a regular basis by an agent called “(OS Administration).” The frequency with which this agent should update the process cache can be set in the application setup document, described previously.

If you already have a process running in your application database, as we do in our example, you have to update the process cache manually if you want to test the initiation immediately on the Web.

Only users with the role “[Process Cache]” in the application database ACL are able to update the process cache manually.

Use the following steps to update the process cache manually:

1. Open your application database.
2. Choose View - Administration - 7. Cache\2. By process.
3. Click Update process cache.

The process cache will now be updated and you will be able to initiate new jobs from your Web clients.

Tip The cache views can also be used to troubleshoot a situation where a process is not displayed for a specific user. In the view Administration - 7. Cache\1. By initiator you are able to see which users can start which processes.

Whenever you make changes to the initiation settings of a process that should take effect immediately at run time, you must update the process cache manually, as described above, after activating the process.

Accessing the workflow database through the standard Web interface

If you performed all the steps described in the previous paragraph, your application will now be accessible for the Web clients. You will be able to participate in the workflow by performing the following steps:

1. Start your Internet browser.
2. Type in the correct URL, for instance:
`http://myserver.lotus.com/myapplication.nsf/?opendatabase`

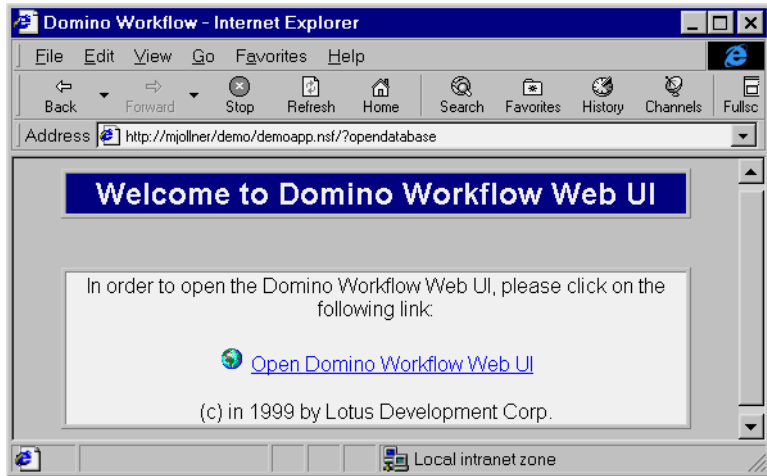
Tip You can also copy the URL from the Setup document, as described above. When you do this the database will open the “All Jobs” navigator and view and the start screen will not be shown. The URL will then look like this:

`http://myserver.lotus.com/myapplication.nsf/OS+WD+A+Name?OpenView`

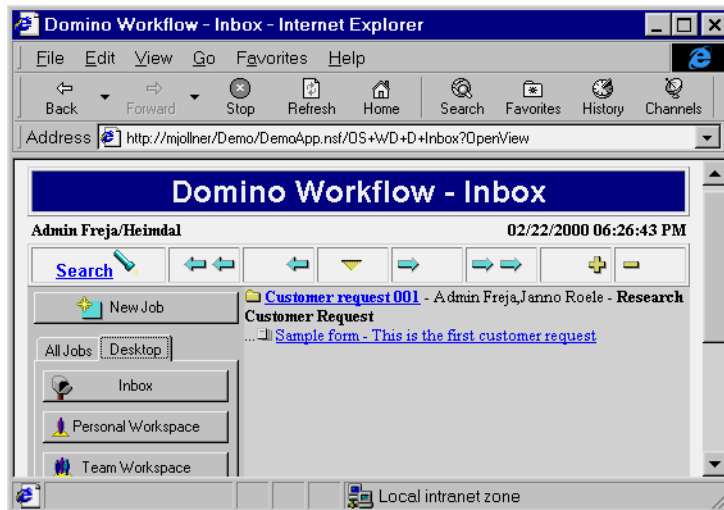
3. Domino will prompt you for a user name and password.

Note The user name should be complete with domain, and the password must be the Internet password registered in the person document in the Domino Directory.

- The start screen of the Domino Workflow Web user interface will be shown. This start screen is the form “\$\$ Navigator Template for All Jobs”. This form is a design element of the application database and, as such, is subject to any changes in wording and code that you may want. You can access its design from the Domino Designer. The default start screen looks like the following figure:



- Click the link “Open Domino Workflow Web UI” to open the database. Domino Workflow opens the default view of the database, which by default is the “Inbox” of the current user. You will now be able to perform the most common tasks, like initiating a job and claiming an activity from your Web client. The following figure shows you the standard Web user interface of Domino Workflow:



The screen in this figure shows the “(OS W D D Inbox)” view within the form “\$\$ ViewTemplate for OS W D D Inbox.” In this form the Web page is built with HTML and Domino functionality (so no additional functionality is used). On this form you can customize your design in terms of changing a title or displaying different information. For instance, you can change the subform “(OS Domino Workflow Headline WEB)” and add your company logo and standard header. This head will be used in all other “\$\$ ViewTemplate” forms.

Shared fields are used to embed references to image files. These references can be changed to your own image files. The image files you want to use must be stored in the directory “domino\icons” or any subdirectory of the directory “domino\icons.”

Note The “domino\icons” directory is a standard subdirectory from the Domino data directory on your server.

The shared fields you can change are listed in the following table:

<i>Field name</i>	<i>Description</i>
DominoDefaultLinesPerView	The number of documents which should be used in the “Next Page” and the “Previous Page” button.
WebBackgroundOS	The background image that is used in the cover document, main document and in all application documents. It is also used as a background image for the “New Job” form.
\$\$HTMLHead	The style sheet that should be applied to all Application Web user interface views. Style sheets are located in your Domino data directory in the subdirectory “domino\html.”
ImageSearchOS	The path and filename of the search image.
ImageMoveFirstOS	The path and filename of the image used on the navigation control that navigates to the first document in the current view.
ImagePreviousPageOS	The path and filename of the image used on the navigation control that navigates N documents backwards in the current view.
ImageMoveMiddleOS	The path and filename of the image used on the navigation control that is used to go to the middle of the view.
ImageMoveMiddleBottomOS	The path and filename of the image used on the navigation control that is used to go to the middle of the view.

continued

<i>Field name</i>	<i>Description</i>
ImageNextPageOS	The path and filename of the image used on the navigation control that navigates N documents forward in the current view.
ImageMoveLastOS	The path and filename of the image that points to the last document in the view.
ImageExpandAllOS	The path and filename of the image that allows you to expand all documents in the current view.
ImageCollapseAllOS	The path and filename of the image that allows you to collapse all documents in the current view.
ImageGoToTopOS	The path and filename of the image that is used in the desktop views to jump to the top of the page.

Advantages and disadvantages of the standard Web user interface

There are advantages and disadvantages to using the standard Web user interface in Domino Workflow 2.0. They are discussed in this section, along with some conclusions regarding the use of the standard Web user interface that Domino Workflow provides.

Advantages

Advantages of using the standard Web user interface include:

- The standard Web user interface is an out-of-the-box Web solution. Domino Workflow ships with its product a standard Web user interface that can be used after making some small adjustments to your application database. You don't need a skilled software engineer to make this work.
- You do not need to change any process-related issues to use the Web clients.

If you are using the Web clients to access your workflow application, the processes you have already developed with the architect do not have to be changed or reactivated.

- Custom changes are fairly easy since the code is all Domino technology and HTML.

Changes to the Domino Workflow standard Web user interface can be made using Domino technology. For instance, you can easily create a frame set where the application should work. Because you use standard Domino technology, customization is possible and fairly simple.

- There is nearly no additional learning curve for the workflow participants because the Web user interface is closely related to the Notes user interface.

If you are familiar with the Domino Workflow Notes user interface, it should not take long to familiarize yourself with the standard Web user interface.

Disadvantages

Disadvantages of using the standard Web user interface:

- The standard Web user interface will, in some cases, not fit into the standard intranet environment your company is running on.

It is very likely that you need to customize the Web user interface before your workflow application can be put into production.

- You cannot access all workflow functionality from the Web user interface.

For example, “add document” functionality, manual join capability, client-based routing, reassignment, and team change are only available through the Notes client.

- The standard Web user interface is very basic, which sometimes affects its user friendliness.

For example, the functionality to complete your activity (Activity complete) is done based on a keyword field where you have to specify the action you want to do on the document and then use the “Submit” button to complete your activity.

- The Web user interface is built based on Domino R4.6. It does not utilize any of the Domino R5.0 Web features.

A lot of Web development features are added to Domino R5.0. These features are not used in the Domino Workflow application because the design is based on Domino R4.6. By adding features such as frames and outlines you can create a better Web user interface for your application.

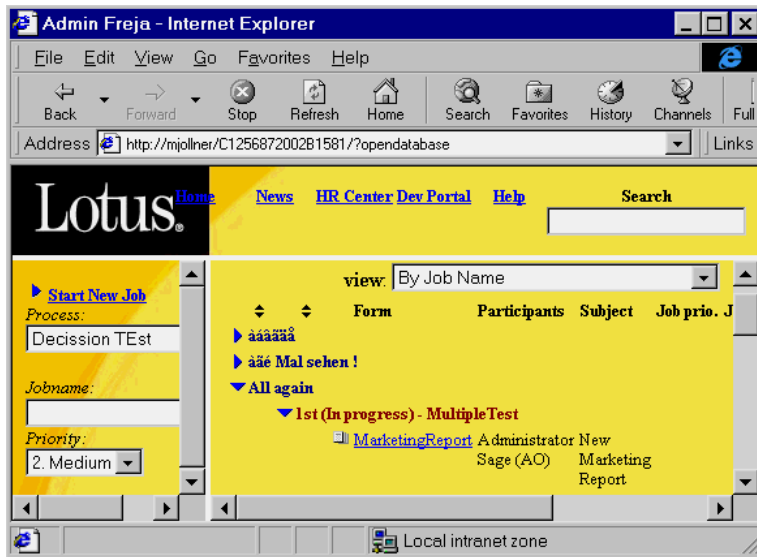
Conclusion

Considering the advantages and disadvantages of using a standard Web interface, we came to the conclusion that it will, in many cases, be necessary to customize your application Web user interface. The customization can be done with standard Domino technology. Therefore, it should be possible for any experienced Domino developer to create a new user interface. By adding R5 features, for instance frames or outlines, you can create a Web user interface which is user friendly and will fit into your intranet environment.

If you only occasionally have workflow participants connecting through the Web, the standard Web user interface of Domino Workflow will be an easy and good solution to use. In such a case you can enable your application for those Web users very easily and without much effort.

Customization of the Domino Workflow Web user interface

The Web user interface shown in the previous figure can be customized according to customer demands and corporate standards. The next figure shows an example of how it can look when you customize it utilizing some of the additional features introduced in Domino R5.0.



A highly customized user interface is shown in this figure. The application is accessed through a frameset consisting of three frames (top, left bottom, right bottom).

The top frame always shows your company's intranet navigator, in this case it is Lotus. In a real-life scenario the workflow application is likely to be embedded in your intranet environment.

The left bottom frame, when first opened, shows the workflow home navigator. From there you are able to initiate jobs. When a specific document is opened, the cover document is visible in the left bottom frame.

The right bottom frame displays Domino views. You can make your choice of one view at a given time. Personal views are customized and the cover document is no longer shown in the view, as compared to the standard user

interface. When a specific document is opened, the binder cover is displayed in the left bottom frame and the main document is opened in the right bottom frame. In the left bottom frame you can also find links to the other documents in the binder. Clicking any of these links causes the specified document to be opened in the right bottom frame.

As you can see, there are a lot of possibilities to customize your Web workflow application. A lot can be done by using Domino technology, including the use of JavaScript. Further discussion of how to implement these changes is presented in Appendix B.

Summary

In this chapter we discussed how to Web-enable your Domino Workflow application. This can be done by adjusting the setup document, updating the process cache, adding a subform to your forms, and enabling the Domino Workflow Web agents with the administrator ID. When these changes are made, workflow participants can access the workflow database with standard Internet browsers and perform most workflow-related actions as they would from the Notes client.

There are some advantages to using the standard Web user interface. You do not have to make any changes and it is an out-of-the-box solution. However, the Web user interface may not fit into your corporate intranet environment. It is also not always easy to use. Therefore you might prefer to customize the standard Web user interface in many cases. Such a customization can be easily done by using the standard Domino features. The details of how to customize your application are discussed in Appendix B.

Chapter 6

Introducing Domino Workflow in an existing application

The goal of this chapter is to show what needs to be done to enable (retrofit) Domino Workflow in an application with a custom-made workflow solution. We will use a standard Document Library database as our example because it features some basic concurrent and sequential workflow. There are several ways to add Domino Workflow functionality to an existing application. In the approach we show in this chapter, we will only use Domino Workflow features introduced in the previous chapters.

General steps to add Domino Workflow to your application

To replace an existing solution use the following steps:

1. Determine the features of the current solution.
2. Create the Organization, Design repository, and Process Definition databases.
3. Define the process in the Domino Workflow Architect, including as many features of the old solution as possible.
4. Add the Domino Workflow design elements to the application database.
5. Customize the Application database to use the Domino Workflow version of the process.

Overview of the process in the existing application

The Document Library is a Domino application that gives teams a place to prepare and share documents and have them reviewed. Besides the content itself, documents have a title, they can be assigned to one or more categories and each document has a review status. Once a document is ready for review, the author selects which people should perform the review. The reviews can be done in serial or in parallel. The author can select to have an e-mail message sent for each completed review or only when all reviews are completed and it is also possible to set a time limit (in days) for each review.

The part of the document form where the review parameters are specified is shown in the following figure:

Originator	Reviewers	Review Options
Soren Peter Nielsen/C&M/Lotus	<input type="checkbox"/> <input type="checkbox"/>	Type of review: <input type="checkbox"/> One reviewer at a time <input type="checkbox"/>
		Time Limit Options: <input type="checkbox"/> No time limit for each review <input type="checkbox"/>
		Notify originator after: <input type="checkbox"/> final reviewer <input type="checkbox"/>

The review process of the existing application has the following characteristics:

- The originator is known.
- There can be any number of reviewers (at least one).
- Review order can be concurrent or sequential.
- Reviewers get a notification for a review activity.
- Notification of the Originator can be done either at each finished review or only at the end of the review process.
- The allowed review time can be specified per document.
- If sequential reviewing is selected, optional automatic submission of a review if it's overdue can be enabled.

Creating the review process

Before we customize the Document Library design, we first have to define the review process in Domino Workflow. This requires us to create a Process Definition database and an Organization database.

Review process definition

We want the new process to cover as many of the features of the review process — as it is defined in the Document Library — as possible. Some of these features can easily be implemented in Domino Workflow, while others require the use of the Domino Workflow 2.0 Developer's Toolkit (see Chapter 7 for instructions on how to get it). However, we will only use features and techniques that were introduced in the preceding chapters, so the process won't be defined exactly according to the current implementation.

New review process features

The new process we will implement has the following features:

- One to four reviewers can be accommodated.
- Concurrent or sequential review order is possible.

- Reviewers get a notification for a review activity.
- The originator is notified at each finished review if sequential reviewing is chosen, and always at the end of the review process.
- A review is overdue after three days.

Organization database

The Organization database has to be populated with process specific elements such as Roles, Persons, and Job Properties. For the review process we will only need a few of the features of the Organization database.

Roles

We will only define the Reviewer role; we will use it to define who may create documents and consequently start the review process.

Persons

The organization database must contain the people in our organization who will be using our new workflow process.

Job properties

We will need access to two fields from the main document.

Note To shield the process definition as much as possible from the implementation details of the main document, it is a good practice to create a Job Property for each field you want to access:

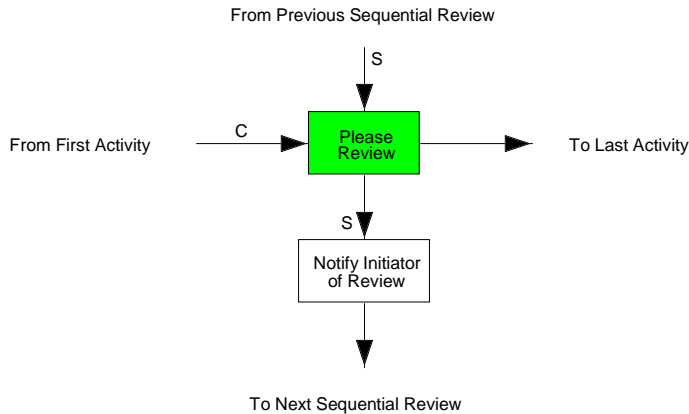
<i>Job Property Name</i>	<i>Formula</i>
List of Reviewers	ReviewerList
ReviewType	@If(ReviewType = "1"; "Sequential"; "Concurrent")

To reduce the complexity of the formulas in the process design we need some additional Job Properties:

<i>Job Property Name</i>	<i>Formula</i>
Select Reviewer	@subset(@subset(@JobProperty([List of Reviewers]) : ""); @input); -1)
Reviewer 1	@JobProperty([Select Reviewer]; 1)
Reviewer 2	@JobProperty([Select Reviewer]; 2)
Reviewer 3	@JobProperty([Select Reviewer]; 3)
Reviewer 4	@JobProperty([Select Reviewer]; 4)

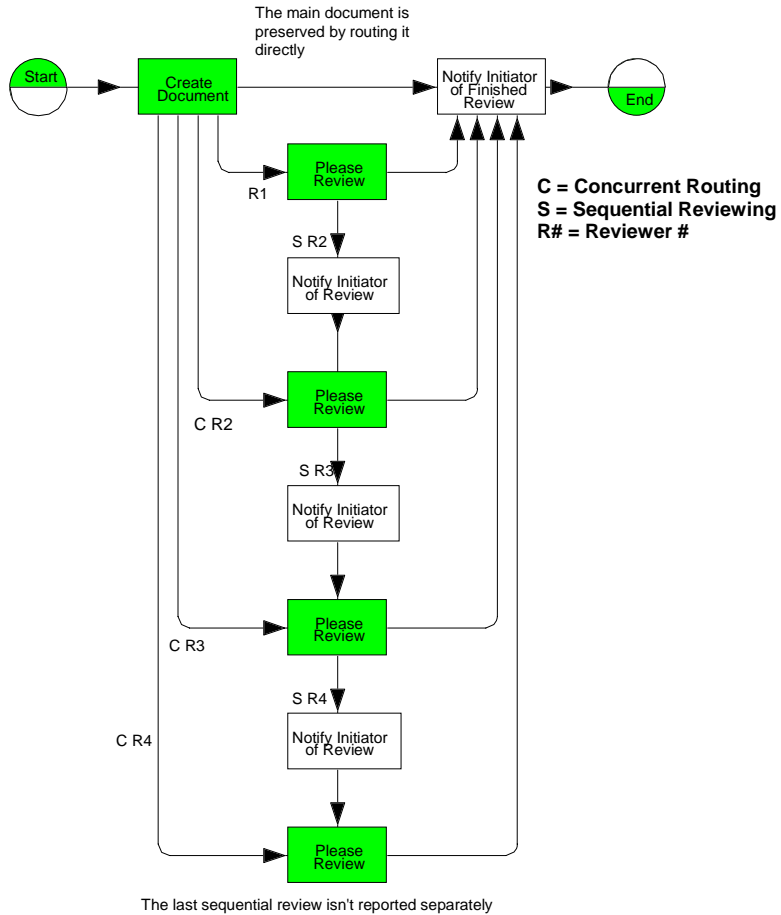
Now we are ready to define the process. We have designed the review process to be used for both the sequential review and the concurrent review. This means that there will be a number of conditional routings in the whole process. The flow is determined by the number of specified reviewers and by the review type.

The recurring element in the process is illustrated in the following figure:

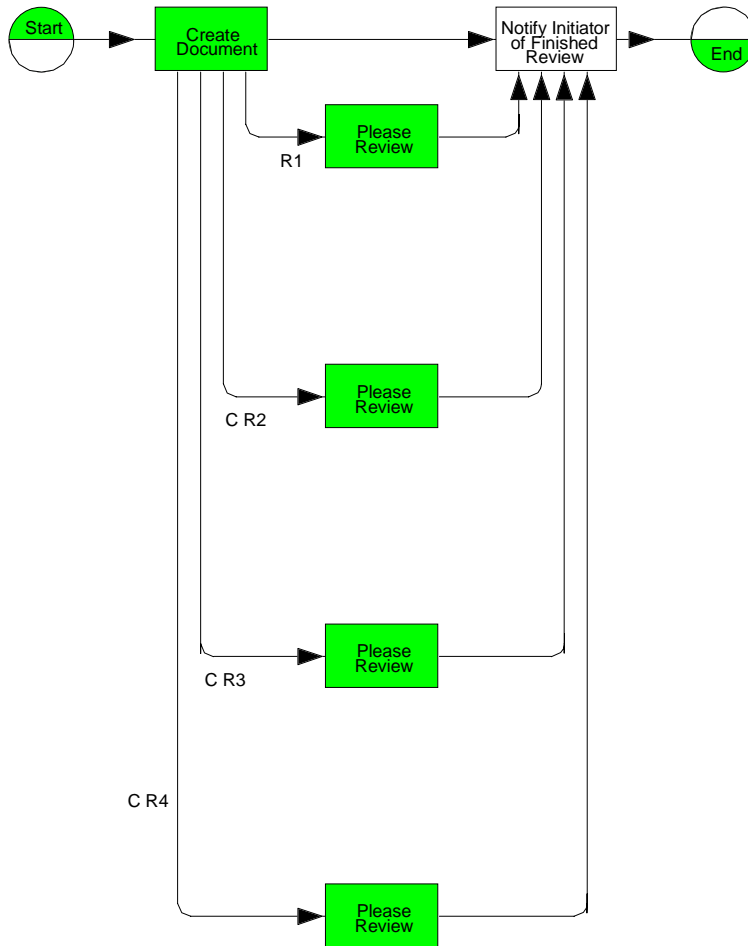


The route from the first activity to the review activity is only taken if Concurrent reviewing is chosen. Both the 'S' routings are chosen only if Sequential routing is chosen.

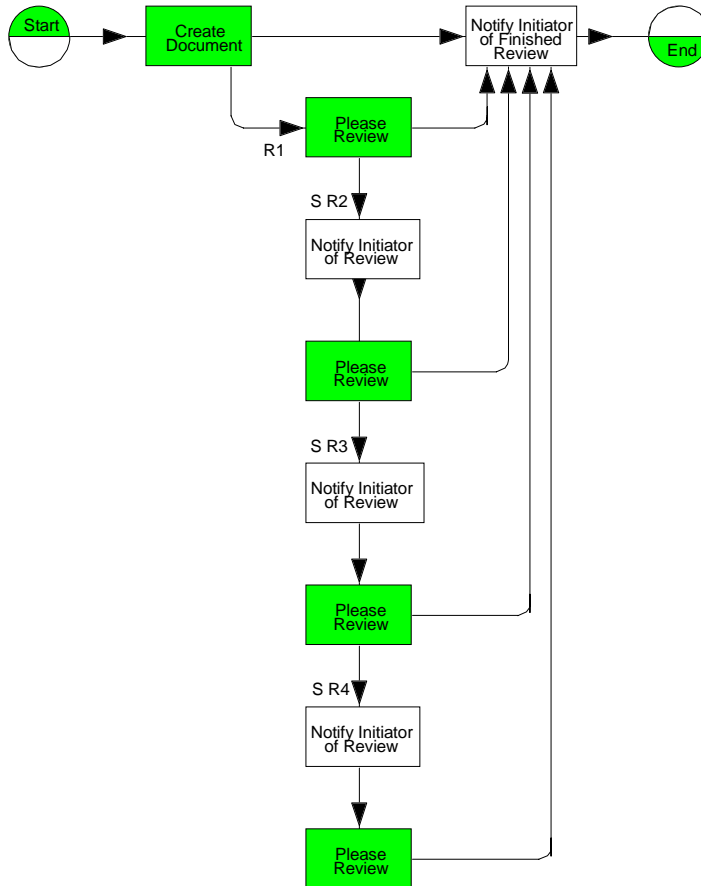
The whole review process, with the maximum of four reviewers, will look like the next figure:



If a four reviewer concurrent process is selected the flow will be as shown in the following figure:



If a four reviewer sequential process is selected the flow will be as follows:



Customizing the Document Library template

Several steps are required to introduce Domino Workflow in the Document Library template, as follows:

1. Add all necessary Domino Workflow design elements.
2. Remove or disable the Document Library design elements associated with the custom workflow implementation.
3. Add the proper Roles to the ACL.

Adding Domino Workflow

Not all the design elements in the Domino Workflow Application template are required. Some are required if you want to participate in a process through a Web client, some are required if you want to integrate with Domino.Doc, and so on. Refer to the Domino Workflow documentation to see which elements you require for your application. The table below describes the required and optional design elements when all users will be using a Notes client (see also the Domino Workflow documentation). The asterisk (*) stands for any sub string, so "(OS *)" matches all names which start with "(OS " and end with ")." Element names which are between square brackets are optional.

<i>Element Type</i>	<i>Notes / Web</i>	<i>Web only</i>
Agent	(OS *) OS * [Show *]	OS Web *
Database Script	Postopen QueryDocumentDelete	
Folders	Search Report	Search Report
Forms	(OS *) (XXX-LanguageProfileTag-XXX)	(OS Web *) \$\$* (OS Binder Cover Web)
Navigators		*
Script Libraries	*	
Shared Fields		*OS \$\$HTMLHead
Subforms	(OS *)	(OS * Web)
Views	Administration\1. Application Setup [Administrator*] [Desktop*] [All Jobs*] (OS *)	*

Removing custom workflow

The visible part of the custom workflow is displayed on the Document Form shown in the following figure:

The screenshot displays a Domino Document Form with the following structure:

Originator		Reviewers		Review Options	
tmpOriginator_1	ReviewerList	Type of review:	ReviewType	Time Limit Options:	ReviewWindow
		Time Limit/Days:	Time Limit (days):	ReviewTime #	
<input type="radio"/> SubmitNow		Notify originator after:	Notify originator after:	NotifyAfter	
Originator		Reviewers			
tmpOriginator	Previous	Current	Future		
	tmpPrevReviewers	tmpCurrReviewer	tmpFutReviewers		

The fields ReviewWindow, ReviewTime and NotifyAfter change from editable to Computed When Composed. The “Time Limit Option” is always “Keep sending reminders after time limit expires.” The “Time Limit” is always four days. The “Notify originator after” is always “final reviewer.”

The actions “Submit for Review,” “My Review is Complete,” and “Clear Review Cycle” are removed.

The hidden field Status reflects where the document is in the process definition. Its value was set directly via actions, but it is changed to a Computed for Display field and it gets its value from the Custom Attribute CA_ActivityAlias.

The field ReviewerNumber was also set via the actions, but was not present on the form. It is added and it gets its value from the Custom Attribute CA_ReviewerNumber.

Adding roles to the ACL

You should add the following Roles to the ACL:

- Install
- Process Reader
- Process Cache
- Process Server

The Install role is used to do the initial application setup. The Language document can only be created by people who have this role.

The Process Reader role is assigned to everybody who should be able to see all documents. People who do not have this role won't be able to see any document.

The Process Cache role is assigned to the administrator group. People with this role are able to update the process cache. The process cache is necessary if the application is to be used via a Web client.

The Process Server role is assigned to the server on which the Domino Workflow agents are enabled. People with this role will be able to run the agent manual, which might interfere with the scheduled agent.

Follow the directions in Chapter 3 to correctly set up the application database.

Summary

This chapter covered the steps necessary to replace an existing workflow solution with a Domino Workflow solution. There will always be advantages and disadvantages when replacing a custom workflow solution with a Domino Workflow solution.

Advantages might include:

- Easy maintenance of an existing process is enabled.
- More complex workflow is possible.
- It is easy to visualize your process.
- Availability of advanced workflow management features such as Time Management or Audit Trail.

On the other hand, there may also be disadvantages:

- Process features of the existing solution might be difficult to design in Domino Workflow.
- The system grows because you need at least two additional databases.

Chapter 7

A closer look at Domino Workflow: process design and implementation

In this chapter we introduce some ways to enhance the process created in Chapter 4. You can use these enhancements individually or in combination. We introduce them separately because you should be able to read and understand any part of this chapter without referring to the other parts.

The enhancements described in this chapter are:

- Using parts of existing designs through the Business Object Library
- Adding job- or activity-specific information to your binder document through custom attributes
- The use of subprocesses to standardize corporate routines
- Various ways to initiate your jobs
- Automatic naming of activities
- Automated activities, that is performing tasks without user involvement
- Automatic claim of activities
- Using parent-child processes to add flexibility to your process

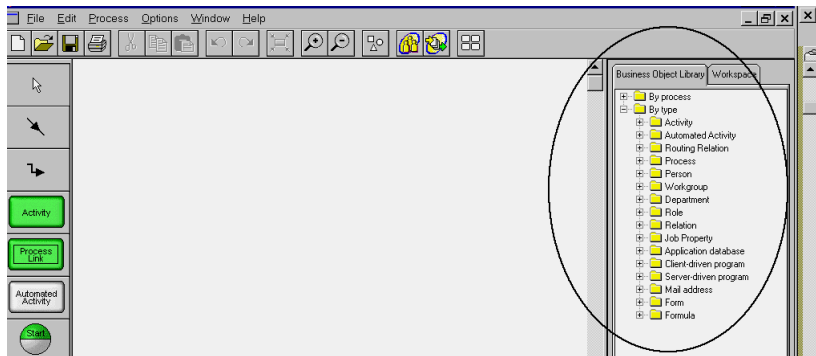
Using existing elements: Business Object Library

The Business Object Library (BOL) is a feature of the Domino Workflow Architect.

It allows you to easily use all the workflow elements that are stored in the databases set in the currently used profile.

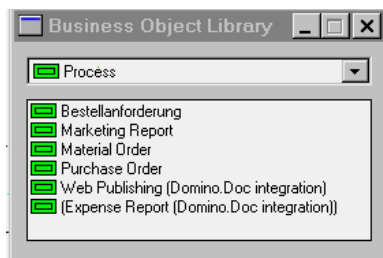
Note You can check which databases are currently open by choosing File - Open databases from the pull-down menu to display the databases dialog. In this dialog you can also change data sources, as well as create multiple profiles for various environments.

You can find the Business Object Library in a split window on the right side of the process window (the window in which you draw your process maps), as shown in the following figure.



Note that in the figure most categories are collapsed. Click on any of the categories to expand it and show the list of all the elements belonging to it.

There is also another way to access a Business Object Library: choose Options - Library from the pull-down menu (or Ctrl L) to create a floating, separate window where you can see all elements for the category you select. One of those is shown in the next figure.

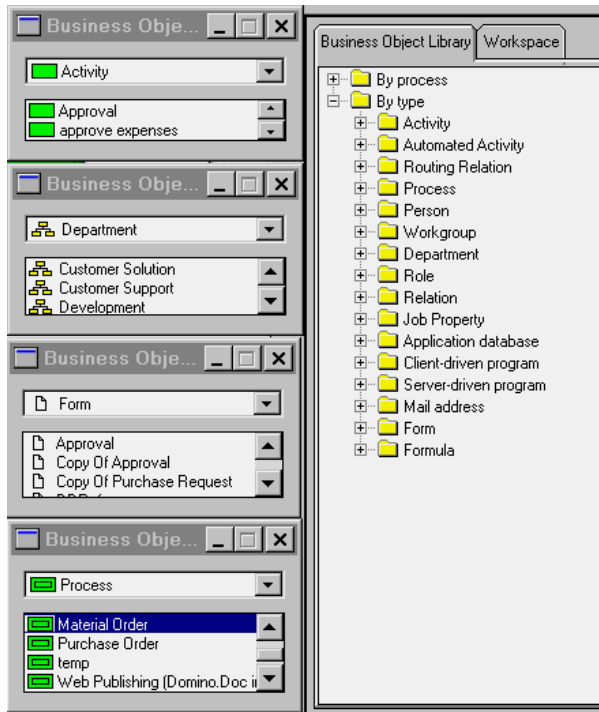


You can open as many windows with the Business Object Library as you want. This is useful for providing access to various object types through several windows without changing the category. The Business Object Library lets you use many different kinds of predefined objects. You can use activities from other processes with all their attributes, as well as automated activities, routing relations, and whole processes.

Note When you reuse a whole process, Domino Workflow creates a process link to the specific process. If this is not what you want because there are some changes in the process, you should open the specific process, rename it, and use "Save as Copy."

You can also reuse formulas that were formerly defined. All those elements are stored in the design repository. From the organization directory there are persons, workgroups, roles, and departments, as well as relations, job

properties, and resources (including application databases, mail addresses, client-driven and server-driven programs) that can be used for modeling. The application database is used to provide the forms that can be used through the process.



You can use the elements of Business Object Library through the drag-and-drop technique. (Choose the element of interest with your mouse, press and hold the left mouse button, drag the element to the place you want to put it, and finally, drop the element by releasing the left button.) You can recognize the permitted and forbidden places to drop your object through the changing shape of your cursor. To transfer elements from one design repository to another, put them in your process window, change the design repository (by choosing File - Open databases from the pull-down menu and selecting the right one), and save your process. When transferring whole processes, the process has to be open in its own window. In case of other references you have to take care of the existence of the referenced objects in the appropriate database.

Note Reusing activities, automated activities, or routing relations always means creating a new copy of the object. Therefore, if you don't want to end up with multiple elements with the same name, rename them for each use. All the other elements used through the BOL are references — you can use them in multiple processes.

Tip Try using your right mouse button for dragging Business Object Library objects. Using this method business objects can be assigned to specific properties of Activities.

Custom attributes

Domino Workflow Architect offers you the opportunity to add temporary fields to the main document of a binder. In Chapter 4 we saw how this feature allowed us to use the same form for two parallel activities and still be able to distinguish between the documents produced because the field value comes from a custom attribute. Custom attributes can be added at the process or the activity level.

Activity level

Domino Workflow adds the fields when a binder is routed to an activity then removes them when the activity is completed. In the Domino Workflow Architect you specify the field name and the value of this attribute in the dialog box for the advanced properties of an activity. For example, you can use these attributes:

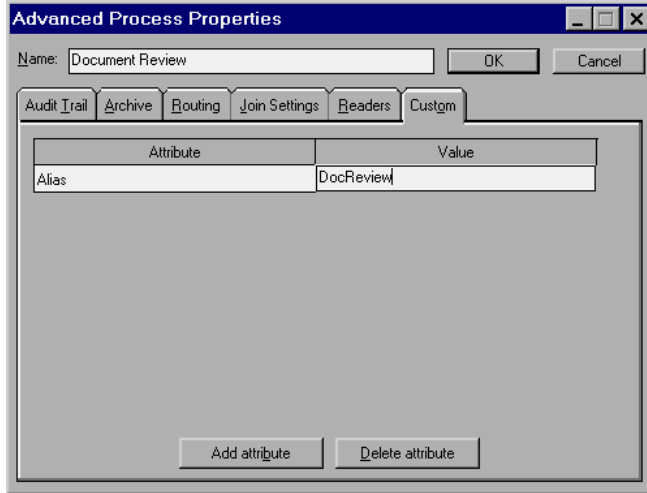
- To pass parameters to an agent in an automated activity. In this way you can use the same agent in different places.
- As activity aliases, so you are not dependent on the full name of the activity or the internal activity identifier to perform actions.

The screenshot shows the 'Advanced Activity Properties' dialog box with the 'Custom' tab selected. The 'Name' field is set to 'Please review'. Below the tabs, there is a table with two columns: 'Attribute' and 'Value'. The table contains two rows: 'Reviewer' with value '1' and 'Activity' with value 'Review'. At the bottom of the dialog, there are two buttons: 'Add attribute' and 'Delete attribute'.

Attribute	Value
Reviewer	1
Activity	Review

Process level

Domino Workflow adds the fields to the main document when a binder is created. In the Domino Workflow Architect you specify the field name and the value of this attribute in the dialog box for the advanced properties of a process.



A custom attribute applied to the whole process is not removed after a job has been completed, while one added at the activity level is removed when the activity is completed.

Standardizing corporate routines: subprocesses

The most important way to reuse processes, as well as run processes in multiple databases, is through the use of subprocesses. Subprocesses are actually just regular processes that are part of some other process. Within the host process you can set them at any place where an activity can be used. You can use a subprocess as some kind of extended activity, with an entrance and exit (point).

A subprocess may also include some other process itself. Therefore, with subprocesses you can create structures of any level of complexity. A process can have more than one subprocess and can be included in other processes, according to your needs.

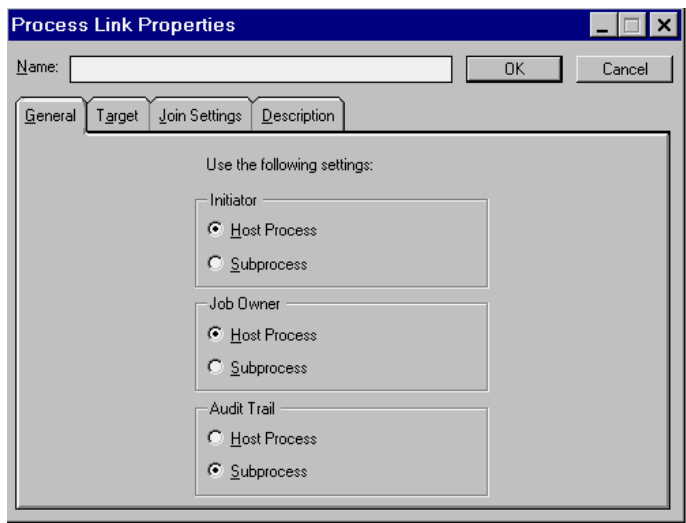
Note Subprocesses are regular processes embedded in other processes. Therefore, you cannot determine from the view of a process if it is used in some other process as a subprocess. In case you should need such information, you'll have to store it on your own (for instance as a Custom Attribute of a process, in its Advanced Properties).

Using a subprocess

The following two methods can be used to introduce a subprocess to your application:

1. Go to the Business Object Library that stores available processes.
Drag a required process to the place you want it to run and drop it there. All basic properties of a process link (that is, what you create using subprocesses) will be set to default values, allowing you to use it as it is.
2. Create an empty link using the Toolbox and manually set the parameters.

The exact description of the parameters can be found in the product documentation. However, you should look at the most important link properties, which are under the General and Target tabs.



In the general settings you can decide, for some of the properties, whether they should be set according to the subprocess or overruled by the settings of the main (host) process. In the target settings you can specify in which application the subprocess should run. You can choose the same application or any other application database, including one located on some other server or even in some other domain. The choice of the application can be based on a dynamic formula.

Note Make sure that the application in which your subprocess will run is connected to a process definition database containing the active process definition of the subprocess. Also, make sure that the application databases are included as resources in the appropriate organization directories and that they are listed as mail-in databases in the Domino Directory.

Note You always have to differentiate between Link Properties and Process Properties. Link Properties are stored in the main process (containing an embedded subprocess), and Process Properties are always stored with the process to which they refer.

Note Domino Workflow Architect also offers an object called Cluster. A *cluster* is used to hide complexity when working in the Architect by representing a subset of the process with a single box. It looks like a subprocess, but the main difference is that a cluster (group of activities) is an integral part of the process and not a reference. Also, a cluster has no attributes other than name and description. It is completely ignored by the engine at the time of process execution.

Customizing Domino Workflow using the Developer's Toolkit

We introduce Domino Workflow 2.0 Developer's Toolkit here because we refer to it when discussing various ways to initiate a process. The information covered in this section should help you to understand some Toolkit basics, to aid in your understanding of other topics.

You can also use the Developer's Toolkit for Domino Workflow R2.0 or later to enhance product flexibility. It is a Domino database with LotusScript libraries, other Domino design elements, and documentation. It contains the workflow events that allow you to extend workflow engine functionality, interfaces to organization directories other than the product's own (as described in Chapter 8), a definition of the XML subprocess interface, and a LotusScript API, as well as definitions of the internal workflow fields. All these elements can be used to customize your own workflow environment.

You can get the toolkit from the Lotus Web site at:

<http://www.lotus.com/workflow>

Important As most elements of the Toolkit are very close to the internal structures and routines used by Domino Workflow, upward compatibility to future versions is not guaranteed. Therefore major new product releases may require you to review customized code extensions. This is especially true once new feature sets are introduced that affect already existing features. However, all the functionality available through the Toolkit will also be supported for future releases. Some custom code may also become obsolete in case a workaround implemented using the Toolkit is replaced by a newly introduced feature. To ease migration we recommend encapsulating the workflow fields and routines into your own functions in order to make the transition easier. Such an encapsulation is always a good practice, as it also makes the maintenance of your application much easier. For more information about encapsulation, see Chapter 12.

Note To use the elements of the Developer’s Toolkit you’ll have to copy some elements included in the database into your application. The exact list of elements you have to copy will depend on your needs and is documented in the Toolkit. Afterwards you’ll be able to use the provided mechanisms.

Important All the workflow events apply to the whole database. Therefore, you have to define the binders appropriate for the action within the event. Such an architecture can also have an impact on system performance since the code will be run every time the event occurs.

Various ways to initiate your job

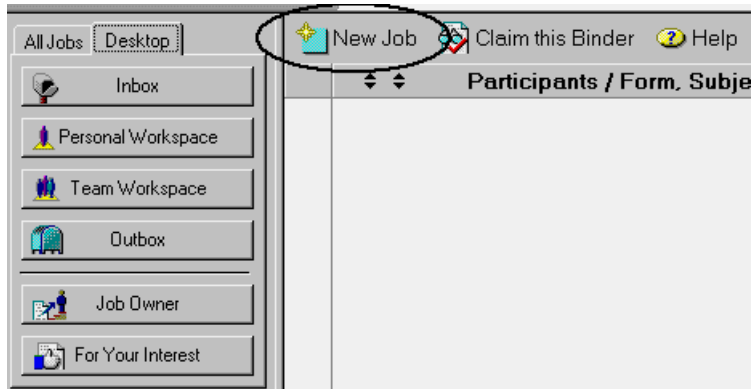
As we mentioned in a previous chapter, there are basically four ways to initiate a job:

- Initiate a job manually using the “New Job” button in your application.
- Initiate a job by incoming mail (mail-based initiation).
- Initiate a job by creating a document in the application database (form-based initiation).
- Initiate a job by using the Domino Workflow Developer’s Toolkit.

These four variations allow you to use the initiation scenario that best fits your needs. Such scenarios include initiating a job from the application database or from any other database (by manual initiation), creating a workflow-initiating document manually, through an agent, or based on a special event (by form-based initiation or the Toolkit). In the next few paragraphs we’ll describe the four basic scenarios in detail. If you need more help on some special features, you can find it in the product documentation.

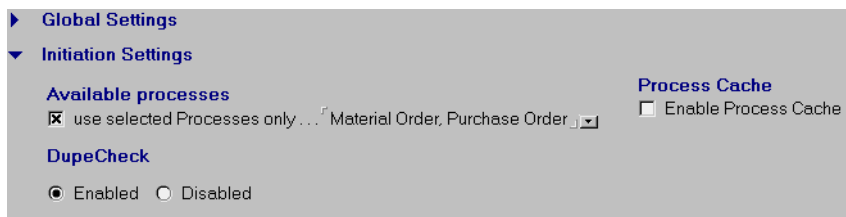
Standard initiation of a new job

The standard initiation can be accomplished from most of the regular views that can be reached using the provided navigator. To initiate a job, click the appropriate action as illustrated in the following figure.



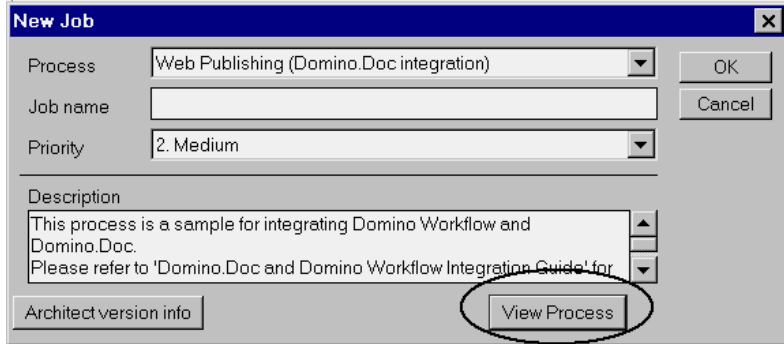
The easiest way to check whether you can initiate a job based on a particular process is by trying to initiate it. If you have any problems finding the right process, there are various settings you can check. They are described in the rest of this section, along with possible settings that you can deploy later on your own.

Note If you want to initiate a job, you must have created and activated a process on which this job will be based. Your application database has to be set on the right process definition database. (You can check it in the application setup: choose View - Administration - 1. Application Setup from the pull-down menu.) You must also be included in the list of people who are allowed to initiate a job based on this particular process. (You can check this in the process definition database, in the Architect, or in the process cache view if you are using a process cache. You can find a process cache view by choosing View - Administration - 7. Cache - 1. By Initiator in the pull-down menu in the application database.) If you've restricted the processes that can be initiated in your application to some chosen ones, you should also make sure that the process on which your job is to be based is allowed to run in this application. You can check this setting in the application setup document, in the initiation settings section as shown in the following figure.



Viewing accessible processes

After clicking New Job, you will be presented with a dialog box for choosing the correct process. If you are not sure which process is the right one, you can read their descriptions and also see the process maps as defined in Architect. Use the Viewer to do this (if you set it up in the organization directory and have been given authorization to use it). Access the viewer by clicking View Process as illustrated in the next figure.



The Web initiation looks basically the same, only without the possibility of using a Viewer.

Advanced initiation

In many scenarios you may want to hide workflow by presenting users with a simple interface, without requiring them to go through workflow-related dialogs. For such purposes you can use advanced initiation. This kind of initiation also can be used in scenarios with remote initiators and for processes that should be triggered automatically. To enable advanced initiation you

have to set the appropriate attribute in the initiation section of the setup document in the application database, as illustrated by the following figure.

The screenshot shows the 'Initiation Settings' dialog box. The 'Advanced Initiation' section is circled in red, indicating that the 'Enable advanced initiation' checkbox is checked. Other settings include 'Available processes' (with a dropdown menu), 'Process Cache' (with an unchecked checkbox), 'DupeCheck' (with 'Enabled' selected), and 'Mail-based Initiation Settings' which includes fields for 'Process' (set to 'Subject'), 'Job name' (set to a formula), 'Priority' (set to 'DeliveryPriority'), and 'Mail becomes' (set to 'Document').

Mail-based initiation

Mail-based initiation takes place when workflow initiators are remote, but can be instructed on how their mail should look or can be given a form on which to send mail. To use mail-based initiation the parameters shown in the preceding figure have to be set. Usage of the parameters is as follows:

- **Process**
The process can be statically named if all mail items should trigger jobs based on the same process. Alternatively, it can be a name computed using an @formula.
- **Job name**
You can use strings, field values, or an @formula to set the name of the job.
- **Priority**
Priority can be set using strings, field values, or an @ formula.
- **Mail becomes**
This parameter defines what should happen to the mail document. It can be deleted, used as a main document in the binder, or put into the binder as a regular binder document.

Note The settings for Process, Job name and Priority have to be defined as formulas. If you want to assign a static string, do not forget to put it in quotes.

After you've set all the parameters and declared the application database as a mail-in database in your Domino Directory, you can begin to initiate processes by sending mail to the database.

Note The main problem with mail-based initiation is to extract a process name and a meaningful job name from the fields available in the mail. Therefore, we suggest using a predefined form that includes fields for a process name and job name to create your initializing mail items or to use form-based initiation. An alternative suggestion would be using only one mail-triggered process for the application database.

Form-based initiation

A similar but more flexible way to initiate a new job is by using form-based initiation. To use this method you must enable the advanced initiation feature in the setup of the application database. Form-based initiation starts a new job after a document containing specific fields has been created, mailed or the fields have been added to an existing document. Therefore, there are various scenarios that can be covered with this technique. It is possible to generate a document manually, or it can be created in any automated way provided by Domino. It's important that a newly created document contain the following fields:

- NewProcessNameOS
- NewJobNameOS
- NewJobPriorityOS
- MailStatusOS
- InitiateOS
- ExternalInitiatorOS

Note These names were used in the 2.0 version of the product. If you're using some other version, check the product documentation for details, in the section "Initiate a Job Using a Form." In the design of the application database there is a form called "(OS Form-based Initiation Template)" that includes the mentioned fields. You should copy and paste these fields into your form and give them the appropriate value.

The field names give a good indication of what you should put into them: the name of the process, the name of the job, job priority, decision about what happens to the initial document, trigger to initiate a job, and the name of the initiator. For further explanations see the product documentation. If you have enabled extended initiation, a new job will be created each time a document containing those fields is created or the fields have been added to an existing document.

Note A new job won't be initiated immediately after you've created a document, but only after the appropriate agent has run on the document. Therefore, you must be prepared for a little delay. The amount of time between creating a document and initiating a job depends on the settings of your agent and your server, and the workload of your server.

Caution The background agent will only create new jobs from documents where the field "initiates" is "yes." The background agent will give this field another value. If you make this field a computed field with the value "yes", every time the document is refreshed, the field will have the value "yes" and the background agent will try to initiate the job over and over again. The background agent will disregard all documents that contain the InitiateOS is "yes" flag, but that are part of a workflow binder already (which means they have a field named FolderIDOS).

Initiation using the DWFJobInitiate API function

If you want full control over when and how new jobs are initiated you can use the API function DWFJobInitiate to initiate a new job. To use this function you have to download the Domino Workflow Developer's Toolkit and install it into your Application database as described in its documentation.

You can use this function from any LotusScript enabled Domino design element such as buttons, hotspots, from a navigator, a form or an agent. All that is required is to add the following statement to the Options section of your script:

Use "OS Domino Workflow API"

Within your script you can then call the DWFJobInitiate function. It has the following list of parameters:

- **ProcessName:** Name of the process that is to be initiated.
- **Jobname:** Name of the job that is to be initiated. There is no check whether the jobname is already in use. Two jobs with the same jobname will not cause any technical problems, but may be confusing for the user.
- **Priority:** Priority of the process. The priority does need to be based on the settings in the application setup document.
- **ClaimActivity:** If true, the binder will be claimed for the current user; if false, the binder will not be claimed.
- **Sideeffect:** When claimbinder is true, the event PostInitByUser is triggered, whereas if it is false the PostInitByMail Event is issued.
- **Maindoc:** Optional: if this variable refers to a document, this document is added to the binder and used as a main document. Use 'nothing' as the default. See also Returnvalues.

- `ErrorMsg`: See documentation.
- `ErrorCode`: See documentation.

The function returns `True` if the initiation was successful. In this case the `Maindoc` contains a reference to the main document of the newly created workflow binder. If the function returns `false`, an error has occurred during initiation. You can use the `ErrorMsg` and `ErrorCode` variables to determine why the initiation failed.

Consider the following scenario: You have implemented a workflow to check candidates that have applied for an open position. One of the options is to invite a candidate to a job interview — which you want to implement as a separate process. At a specific activity you want the Activity Owners to be able to start this new process manually using a “invite to interview” button. All information about the candidate is to be transferred to the new workflow.

You can implement this scenario by creating a button on the form used by the first process. In its Options section the button contains the statement:

Use “OS Domino Workflow API”

In the Click event the following code is being used:

```
Sub Click(Source As Button)
    Dim ws As New NotesUiworkspace
    Dim session As New Notesession
    Dim thisdoc As Notesdocument
    Dim newdoc As Notesdocument
    Set thisdoc = ws.currentdocument.document
    Set newdoc = _
thisdoc.CopyToDatabase(session.currentdatabase)
    ProcessName = "Invite candidate"
    Priority = "2. Medium"
    JobName = "Invitation for " + thisdoc.CandidateName(0)
    Claim = True
    success = DWFJobInitiate(ProcessName, JobName , _
Priority , Claim, Newdoc, ErrorMsg , ErrorCode )
    If success Then
        Call ws.EditDocument(True, newdoc)
    Else
```

Msgbox Errormsg

End If

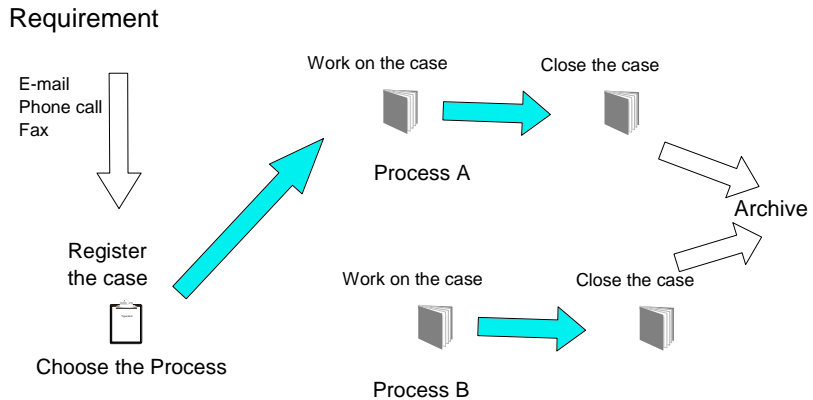
End Sub

Using the Domino Workflow API function DWFJobInitiate is the most powerful way to initiate new jobs programmatically.

Note Depending on the version of the Developer's Toolkit that you are using, the parameters may be slightly different. There might be an additional input parameter "OnBehalfOf" that allows you to claim the first activity for someone other than the current user. This option is useful, if you are using the DWFJobInitiate function in a Web agent running on the server's name. Refer to the documentation for more information.

A practical initiation scenario

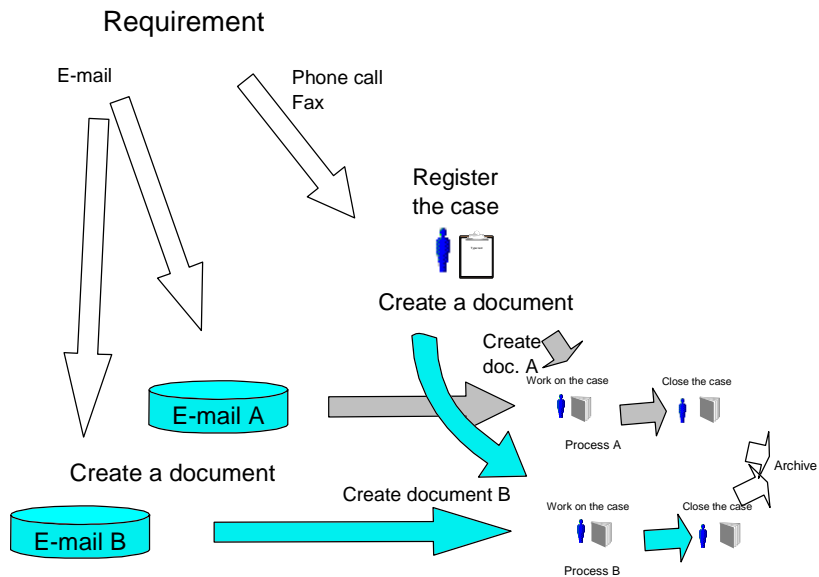
Consider the example of two simple processes running in one workflow application. Both of them can be triggered manually (if a customer calls) or by e-mail. There is no possibility of influencing the content of customers' mail; you can only give them two different addresses for the different processes. This scenario will look like the following figure.



You can use a manual initiation exclusively for all incoming requirements, or you can develop some structure to automate a part of the initiation.

The first step would be to create two forms that will trigger your processes — let's name them form A and form B. Both of them must include appropriate fields allowing the creation of a job. In this way a person registering one requirement can create a document based on an appropriate form to trigger the right job. To automate creating jobs based on an e-mail, you can create two mail-in databases that will act as filters for the incoming mail and trigger the right process. An agent in each of these databases will create a document in the application database.

In this scenario the dispatchers don't actually work on the case, but only register it by using the appropriate form. They don't have to be aware of the workflow following their action, as all the actions they will take are typical for Domino. The external initiators also don't have to know anything about workflow other than an e-mail address; they send mail with any kind of subject and content on the appropriate self-explaining address (for instance, bugs@lotus.com and ideas@lotus.com) that are then automatically processed by the dispatcher. You can see such a scenario in the following figure where EmailA is a database, EmailB is a database and Process A and Process B take place in the same database C. All "create a document" actions happen within database C.



Realization of the scenario

As we already mentioned, you will need two active processes and two corresponding forms. Both forms must include the six fields required for form-based initiation. You have to put the appropriate process names into the field `NewProcessNameOS`: use the exact name of process A in form A and the exact name of process B in form B. Then choose a name pattern for the jobs. We suggest a name pattern that is at least partly computed and unique for each job. This will allow you to identify your jobs later on.

Note You can enforce unique job names by setting the parameter `Dupe Check` in the application setup as `Enabled`. If you do this, you must always provide unique names for the automatically initiated jobs. Otherwise, new jobs can't be initiated and the application owner will get an error message.

As a second step you should create two mail-in databases that will receive incoming mail.

Note You may also use one database with two different e-mail addresses and then select mail by the address, but it's basically the same scenario.

Next, create an agent in each database that does the following things:

1. Create a new document in the application database.
2. Assign either form A or form B to this document by creating a field Form with a value of A or B (name of the appropriate form) according to the required process.
3. Set the values of all fields required for form-based initiation on the new document with the following values:
 - NewProcessNameOS as process name (A or B)
 - NewJobNameOS as job name (according to the instructions given in the beginning of this section, for example using the subject value of the incoming mail)
 - NewJobPriorityOS as priority according to the rules used in your company
 - MailStatusOS as an empty string to use your newly created document as a main document in your job
 - InitiateOS as yes to allow initiation of a new job
 - ExternalInitiatorOS as the name of the author of the mail
4. Create fields on the new document that will contain the Subject and Body of the mail and copy the appropriate values into these fields.
5. Save the new document.

This agent should run on all new documents in the database to be sure that each mail creates exactly one job in the application database.

Note The most logical trigger for such an agent would be After New Mail Has Arrived, but you may experience some problems with it, as in some versions of Domino the scheduling of such agents doesn't work properly. If you encounter such problems, set your agent to a schedule.

To enable your new agents, communicate the appropriate mail addresses to all potential authors of incoming requirements and the jobs can be created — manually or automatically.

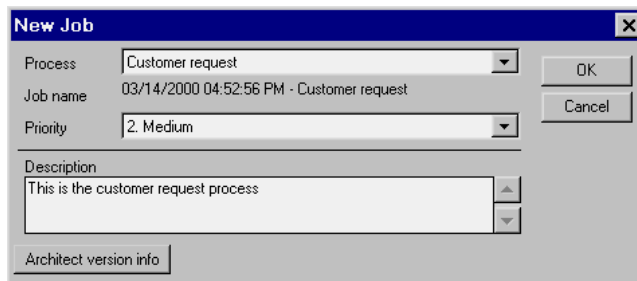
Naming your job automatically

Creating meaningful names for every job is not an easy task for end users. You may already have some naming conventions for specific jobs. In such situations it is easier to create job names automatically rather than manually. You can do this easily with Domino Workflow.

By a standard initiation

When initiating a job with the “New Job” button, Domino Workflow creates a dialog box based on the form “(OS Dialogbox Init).” In this dialog box you must fill in the specific process you want to start, the job name you want to give to the job and the priority of the job. By opening the “(OS Dialogbox Init)” form in Domino Designer, you can change the fields on this form into computed fields and give them a computed value. In the following figure the field “JobNameOS” is changed to a computed field with the formula:

`@Text (@Now) + " - " + ProcessListDisplayOS`



Creating your own computed job names helps you to provide more consistency in views and to better present the jobs to the end users. It can also save time during job initiation, as the users won't have to take care of the unique and informative naming.

Using workflow events

When you create new jobs using alternative initiation methods, it may be a good practice to generate job names automatically according to a specific scheme. While you can reach this goal by attaching some logic to your form or database, it actually isn't the right place to do it. The job name is a part of workflow and, as such, it should be created as a workflow task. The only problem is the fact that workflow needs a job name; therefore, it has to be created before the workflow starts. This is impossible to do using a standard product, but Domino Workflow Developer's Toolkit offers a QueryInitByMail event that takes place exactly at the time we need it — when an alternative initiation takes place, but before a job is created. Before using it, some scripts have to be added to your application from the Domino Workflow Developer's

Toolkit, according to its documentation. As an example of what you can do, we show a piece of code along with code from the Toolkit.

Note The following piece of code uses the methods `GetResource` and `Evaluate`, which you will have to implement in order to make this code work.

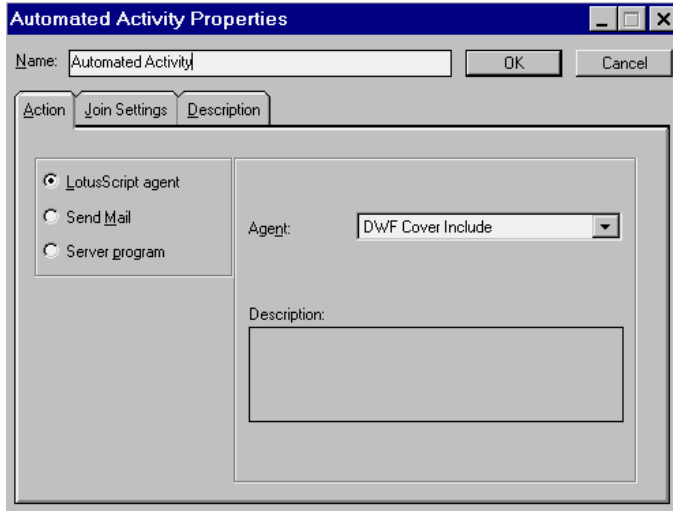
```
Sub QueryInitByMail(Continue As Integer, Mail As
NotesDocument)
' Variable declaration
    Dim objSession As Variant
    Dim objResource As Variant
    Dim vReturn As Variant
    Dim docCurrent As NotesDocument
' Find resource in which your job name will be calculated
' It may be a central resource database
    objSession = OpenSession ()
    Set objResource = objSession.GetResource
' Set resource -- evaluate your resource
' to get the process name according to your schema
' and set the field value of NewJobNameOS to this name
    Call objResource.Evaluate(docCurrent,Null,vReturn)
    Call Mail.ReplaceItemValue("NewJobNameOS", vReturn)
End Sub
```

What you can achieve using such code for creating a job name is the possibility of keeping all possible job name schemes in one place, in order to bring more structure into your work.

Basics of automating an activity

Using an automated activity is an easy way to help workflow participants do their jobs by transferring some of their highly repetitive and standardized work to the server. Automated activities act on the job context represented through a binder, and therefore can react to the changing situation according to the rules you have given them. The standard set of possible actions that

can be taken via automated activity includes sending an e-mail, triggering an agent, and executing a program on the server.



Sending a mail message

The simplest action that an automated activity offers is sending a mail message. You can send it to any person or group defined in your organization directory. You can also use job properties to define an address, or you can use a predefined address or formula that extracts some information from the job content to specify who should receive the mail. For a subject, you can use any static string or formula for context information. The body can also be composed from some static information, and all required content extracted from a job through your custom formula. To help you with creating the formula there is an assistant that provides all the available fields and properties.

Note Keep in mind that all parameters have to be defined as @formulas. Don't forget to use hyphens when entering text statements.

Executing a server program

Using this option you can trigger an executable on your server. Such a server program has to be previously defined in the organization directory as a resource.

Caution If you use this option, be sure that only appropriate people have edit access to the organization directory. Otherwise, you run the risk of breaching security by running an executable on your server machine that could be manipulated by another person.

It is also possible to pass parameters to the executable. Those parameters can be of static nature, or they can be computed using formulas that extract information from the job content.

Starting a LotusScript agent

The most powerful possibility for an automated agent is executing a LotusScript agent. You can use the automation template delivered with the product to create an agent, as described in this section.

Note You have to make sure that CustomSubroutine in the template is changed to return True. Without this change the automated activity will never be completed.

Important You have to be sure that your agent was enabled (signed) by somebody with the authority to execute unrestricted agents on the server. Otherwise, execution of the agent will be impossible — and the only place where you can see it is in the server log. Domino Workflow sends no notification at such a point, as the problem is related only to server access rights.

Developing a LotusScript workflow agent

The template for your agents used in the automated activity is an agent called (OS Automation Template). You should make a copy of this agent and name it according to your rules and needs. Your agent name must not start with “OS,” “DWF,” “(OS ” or “(DWF” as agents starting with these letters will be regarded as Domino Workflow system agents and are not listed by the Domino Workflow Architect.

This agent allows you to access the parts of a binder through the variables described in the function, CustomSubroutine. Those variables are:

binder

Contains a NotesDocumentCollection of all Binder Documents (including Cover and Main document).

cover

Contains a NotesDocument, referencing the Cover Document.

main

Contains a NotesDocument, referencing the Main Document (if you use parallel paths, the binder may contain several main documents).

The function CustomSubroutine should be extended with your own LotusScript code. The result of your code should set the value of the variable “CustomSubroutine.” In order to route the binder further in the process after executing your code, it is crucial that the CustomSubroutine

variable is set to True. This will happen when you put the following line on the end of CustomSubroutine, just after your code:

```
CustomSubroutine = True
```

If you want to temporarily disable further routing of the binder, you can set:

```
CustomSubroutine = False
```

This will protect the binder from being passed to the next activity. You can picture scenarios where your script decides whether the binder should be routed or held. When the CustomSubroutine is False, the background agent will evaluate the code every time it runs, until the CustomSubroutine is True.

An example

Consider a scenario in which you want to use an automated activity to retrieve employee information from your human resources system. The Custom Subroutine function of your Automated Activity Agent may look like this:

```
Function CustomSubroutine (binder As OSDocumentCollection,  
cover As notesdocument, main As notesdocument) As Integer
```

```
    On Error Goto errorhandler
```

```
    Dim EmployeeID As String
```

```
    Dim EmployeeData As Variant
```

```
    Dim success As Integer
```

```
    EmployeeID = main.EmployeeID(0)
```

```
    success= GetEmployeeDataFromHRSsystem(EmployeeID,  
EmployeeData )
```

```
    If success Then
```

```
        main.Salary = EmployeeData(0)
```

```
        main.Manager = EmployeeData(1)
```

```
        main.Office = EmployeeData(2)
```

```
        CustomSubroutine = True
```

```
    Else
```

```
        Error 20000, "Data cannot be fetched"
```

```
    End If
```

```
    Exit Function
```

```
Errorhandler:
```

```

    ' implement a function to report the error
    CustomSubroutine = False
    Exit Function
End Function

```

When creating your own Automated Activity agents you should keep the following recommendations in mind:

- Always implement error handling. Within each routing cycle the agent is only called once and processes all activity instances that are assigned to this agent. If the agent is aborted when processing the first workflow binder, all other workflow binders that need to be processed by this agent will also be skipped, even though the error that causes the agent to abort may only occur in very specific circumstances.
- Do not use any NotesUI classes. As automated activities are run on the server, UI classes cannot be used and will prevent the agent from executing.
- Automated Activity Agents are started by the OS Background agent using the agent.run method. This has a side effect; prints will not appear in the server log. You may use the NotesLog class or another protocol mechanism instead.
- Debugging Automated Activities may require some workarounds.

Since Automated Activity Agents are automatically triggered by the OS Background agent, you are not able to debug them using the LotusScript Debugger. However, there is a workaround that you may use. You can prevent the agent from starting automatically either by using client based routing or by renaming the agent after activating your process. If you then start a new job and route the binder to the activity in which it will be processed by your agent, the binder will have the status "Automation." You may copy the binder to the clipboard in order to debug the code several times. If you have renamed the agent, undo this change. Now you can run the agent from the agent menu, with the LotusScript Debugger turned on. If the CustomSubroutine function is not called, the agent cannot find any workflow binder to be processed. This is usually related to a change in the agent name. If the CustomSubroutine is called you can debug your code. If you want to prevent the agent from actually routing the activity (in order to debug it again), abort the agent, before the CustomSubroutine is completed. To repeat your debugging session, you can also delete the original binder (assuming you have the access rights to do so) and paste the binder that you have copied to the clipboard back into the Application.

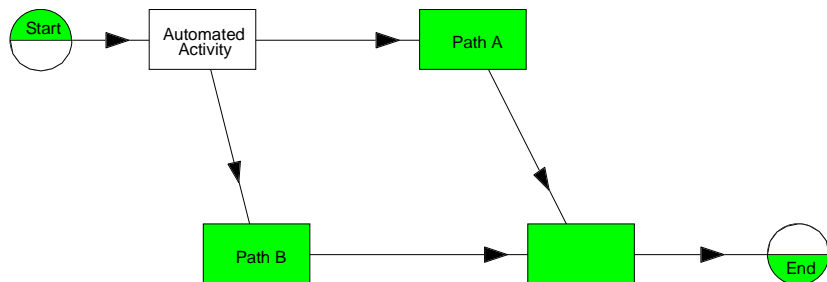
When debugging an Automated Activity Agent, bear in mind that you are debugging in a client environment, whereas the agent is executed on the server in the operational environment.

Additional scenarios using an automated activity

Although the techniques used to create automated activities were already covered, there are also some enhancements to Domino Workflow that can be done. In the basic version of the scenarios that we introduce here, it's enough to create an agent based on the automation template by setting the variable CustomSubroutine to True, as described previously. Then you can call this agent from the automated activity placed in the appropriate place according to the scenario, and the rest will happen automatically. You can put additional code into the agent if you want to enhance its functionality, but we strongly recommend trying the scenarios with a very easy agent at first to eliminate possible process errors.

Automatically created jobs with multiple entrances

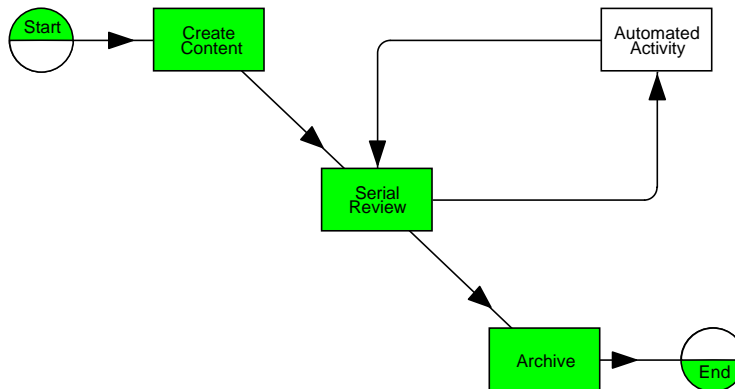
There are some cases where you have a process with more than one possible first activity. This may happen, for instance, if some of the incoming requests were already categorized by the sender and some others were not. You would categorize them in the first activity, but those that are already categorized shouldn't enter this activity at all. Since a process can have only one entry point, there is only one way of solving this problem. You can start your process with an automated activity that will represent only a starting point for the process, without any further functionality. To do so you should let this activity trigger an agent based on the automation template that evaluates to True (as described previously in this chapter). Then you are able to create multiple paths exiting the first activity and routing a binder to some other activities according to the rules defined in the routing relations.



Such a situation happens rather often. You could also try to resolve it using embedded processes, subprocesses, parent-child processes, or some other workarounds, but we consider this solution to be the most efficient, quick, and elegant one.

Loops to the same activity

There are some activities that have to be repeated many times, each time evaluating the new group of potential activity owners. Such a situation might happen, for instance, in the serial review of a document. There are various ways of modeling such a situation, but the easiest and most comprehensive possibility involves using an automated activity. Such an activity executes only a dummy agent based on the automation template and its evaluating to True, as described previously. Then you should define a condition on which the loop will be activated (it may also be an explicit decision of the activity owner) and you can run your process. At every loop the potential activity owners will be newly evaluated, and might change according to the logic included in their definition on the process level (in Architect). Make sure that your loop will come to an end at some point, and the binder is routed to another activity. Otherwise your job will never terminate.



Note You may also put some custom code in the agent triggered from the automated activity. It's a perfect place to move some evaluating logic from your form to a place where it'll be more obvious and easier to document.

More information about automated activities running LotusScript agents can be found in Chapter 12.

Automatic claim of activities

In Domino Workflow, a job always has to be claimed by one of the potential activity owners before they can work on it. However, there are many situations in which such an approach slows things down and is not appropriate. For example, you may have only one potential activity owner. In such a situation, you can automate claiming of the binder and move it directly to a status in which a workflow participant can start to work. You can develop such a mechanism by using the Domino Workflow Developer's Toolkit. Write an agent acting on the documents and use the function

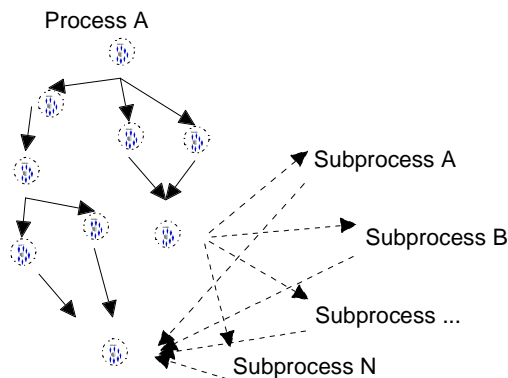
DWFActivityClaim to change the status of the binder from New to In Progress and to transfer the binder into the In Progress view of an activity owner. This function is a part of the Domino Workflow API, and is described with all its parameters in the Developer's Toolkit product documentation. After you add the required script library from the toolkit into your application database ("OSApplicationEvents"), you can use it everywhere inside your application. Therefore, it's also possible to use this function in the event PostFolderRouting that also is included and documented in the Toolkit, as well as in an agent. In this case, it is possible to enable automatic claim for all activities that have only one potential activity owner.

Note The workflow events refer to all jobs in the application database and are executed for every process. Therefore, you may need to include an if or select statement within your event code to limit the functionality that you have added to a specific process, activity, or scenario. Often it is useful to create custom Job or Activity Properties in the Domino Workflow Architect in order to be independent of Activityname or Jobname changes.

Supporting a family: parent-child processes

In many situations it is necessary to create one or more jobs based on some dynamic parameter or event. If the process type doesn't change and the number of jobs that should be created is stable, you can do this through subprocesses. However, if any of those conditions do not fit, you run into trouble. It is not possible to handle such a situation using only a standard model of Domino Workflow. You have to extend this model with your own development. The possible scenarios that require such an effort are:

1. Unknown number of related processes (depending on dynamic situation).



An example of such a situation is the review cycle of some marketing materials, where for less important materials up to five reviewers exist; but in the case of strategic materials all sales managers (20, 30, maybe more) are also asked for a review.

2. Processes based on just a part of the content of the host process.

An example process related to marketing material production can be used, in which various parts of the material should be reviewed by different people. The reviewers shouldn't be informed of the existence of other parts of the material.

3. Processes that don't return any information to the host process.

In this category are all processes that don't relate to the parent process after creation.

In all those scenarios you can start the child processes through a form-based initiation. The appropriate documents can be created by an agent which will be triggered from the automated activity. For the most complex scenario — the one with an unknown number of subprocesses that have to be coordinated with the main process — you can find a detailed description and the required code in Appendix A.

The basic idea of this scenario is:

Place an automated activity at the place in your process where you want to generate some new processes. Create an agent that takes care of the following actions:

1. Create a field in a binder of a parent process to register all created documents.
2. Create and register the required amount of documents for initiating new processes in the appropriate databases.
3. Add the fields required to initiate a job to those documents.
4. Transfer the appropriate content into the newly created documents.
5. Transfer parent reference information to the child processes.
6. Set binder status so that the binder waits for all child processes.

You also have to create a mechanism for the response from the child processes. They have to provide the required information to the parent process, check themselves out from the parent process, check if there are any more children left, and (if there are no more children) set the status of the binder to the one that allows further routing of the binder. For the code used in the agents and an example look in the referenced appendix.

Summary

In this chapter we've tried to show you some more features of Domino Workflow that can help you to automate your processes. We've described some standard possibilities for workflow deployment, as well as some ways requiring more complex customization. There is much more to workflow than we were able to cover here, and your company will probably need some features and customization that we didn't cover. We only hope that we can help you to make the first steps that are common for most installations. You'll still need all your Domino skills to build a perfect environment for the work that happens in your company, of which workflow is only a part.

Chapter 8

Using the organization directory

The organization directory is an important part of your workflow environment. Domino Workflow has its own organization directory database where you can define persons, departments, workgroups, roles, surrogates and out-of-office profiles. Besides the organization data, it is possible to store system-specific information in the organization directory, such as relations, job properties, application databases, client- and server-driven programs and mail addresses. In this chapter we discuss the use of the system-specific information. The other functions of the organization directory were presented in Chapter 3.

In addition to using the organization directory supplied by Domino Workflow, many organizations already use the standard Domino Directory to store their organization data. Furthermore, some organizations have developed custom extensions to it, for instance, the Lotus Solution Architecture (LSA) organization directory. In this chapter we discuss the integration of these other organization directories. We identify some of the advantages and disadvantages of using another organization directory and describe how it can be done in practice.

Why use a separate organization directory

In every organization, and especially in those using a workflow approach to execute business processes, people are the most important factor. Your workflow application will depend on the actions taken by people within your organization. In a Domino environment every person is registered in the Domino Directory as a certified server user. In addition, you are able to define groups in the Domino Directory where you can group certain people and organization units. Domino Workflow offers some additional functionality to the Domino Directory in order to use it in the workflow environment. This additional functionality allows you to automate some highly repetitive tasks and decisions. Therefore, you can concentrate on other important parts of your work.

Note All the features used in the Domino Workflow organization directory can be implemented in the Domino Directory by customizing your Domino Directory. However, in a lot of organizations the customization of the Domino Directory is not allowed. This is another reason that Domino Workflow created its own organization directory.

The following functionality, which is used in Domino Workflow, is not offered in the Domino Directory:

- Organization hierarchy

In the Domino Directory you are able to group people and thus create a hierarchical structure. However, this hierarchy cannot be displayed in Domino Directory views. You are also not able to define the manager of a certain group. In Domino Workflow, this manager property can be stored in the Department and Workgroup documents and used by the workflow engine to send notifications to managers, as designed in Architect (by the use of a job property which will be described later).

- Out-of-office and surrogation profiles

When an organization unit is not functional because all members are out of office, Domino Workflow is able to route the activities to “surrogates.” By doing this, your process will not be delayed. The surrogates are defined in the participant, workgroup, department, or role documents; and the out-of-office profiles are created in a special view, where all out-of-office profiles are stored.

- Organization-wide roles

In the organization directory you can create role documents, which you can use in multiple applications.

- Resource storage place

Besides offering advanced hierarchy to your organization, the organization database is also a storage place for system-specific information. You can define relations, job properties, and resources such as mail addresses, application databases, and client-driven and server-driven programs. This system-specific information is the main subject of this chapter.

Domino Workflow recognizes the Domino Directory as the basic certification and storage place; therefore, it offers functions to import information from the Domino Directory into the organization database.

In many organizations, the use of another organization directory will not be accepted because you will have to maintain your organization data in two places. Therefore, Domino Workflow allows the integration of the Domino Directory and other Notes-based organization directories, like LSA Orga, in their applications.

Domino Workflow also stores information that is specific to a Domino Workflow environment and that will not be available in other directories. Examples are relations, job properties and resources. Therefore, the Domino Workflow Directory can be used as a Resource Database even though the organizational information is stored in another directory.

Note You need the Domino Workflow organization database as an organization directory (also containing resources) or only as a resource. The second case (organization directory as resource storage place) takes place if you're using an alternative organization directory, such as the Domino Directory.

There are advantages and disadvantages to using either the Domino Workflow organization directory or the Domino Directory. Some of them are summarized in the rest of this section.

Using the Domino Workflow organization directory

Advantages

- The Domino Workflow Organization Directory is built to host an organizational model for workflow applications. This includes features such as managing hierarchical department structures, workgroups, surrogation rules and Out-Of-Office Profiles.

All structures can be defined independently from other directories, especially the Domino Directory. This allows creation of a group structure that explicitly suits workflow needs. It also allows the setup of Domino Workflow at a departmental level without affecting the general Domino Infrastructure.

Disadvantages

- You have to maintain your organization directory information in two places.

You have to maintain the Domino Directory (or another directory database) and the Domino Workflow organization directory. In small workflow applications this should not be a big problem, but in larger applications this may lead to inconsistency problems. It also creates additional administrative overhead.

Using an alternative organization directory

Advantages

- You have to maintain only a single organization directory.

By integrating your current organization directory with the Domino Workflow application, you have just one place where your organization is maintained. There won't be any concerns about inconsistency between two organization databases and you will be able to use the full functionality of your current organization directory.

- When setting up your workflow infrastructure, it might be easier to set up Domino Workflow to use the existing organizational information rather than recreating this information in the Domino Workflow Organization Directory. This applies if you already have a well managed organization directory.

Disadvantages

- When you use another organization directory you will lose functionality in comparison with the Domino Workflow organization directory.

For instance, the Out-Of-Office profiles, which can be used in the Domino Workflow organization directory, cannot be used in another organization directory. However, you can implement the missing functionality yourself.

Conclusion

As mentioned before, there is no simple answer to the question of whether you should use the Domino Workflow organization directory or another Domino-based organization directory like Domino Directory or LSA Organization Directory. For every specific situation you should consider the advantages and disadvantages of your decision.

Advanced use of the Domino Workflow organization directory

In this section we describe the advanced capabilities of the Domino Workflow organization directory. There are three main advanced issues in the organization directory, as follows:

1. Relations

Relations are formulas that deliver a particular item of information, given specific search criteria. For example, Francis is the “Secretary of” Tim, or Debora is the “Girlfriend of” Janno. In these examples Francis or Debora are the search criteria and the particular item of information will be Tim or Janno. You will find more about relations later in this section.

2. Job properties

Job properties are formulas that represent information generated during the course of a job, and that is therefore not known during design time. For example, “Activity owner of activity,” means the person (belonging to the group of potential activity owners) that took the responsibility for a given activity in a particular job. During the job, each activity owner is stored in a specific field. If you want to know the activity owner of a specific activity you can use this job property. You will find more about job properties later in this section.

3. Resources

A resource is an identifier to a specific aspect of your company’s information technology infrastructure. Resources can be either application databases, mail addresses, client-driven programs, or server-driven programs.

Relations

The definition of relations is: “Relations are formulas that deliver a particular item of information, given specific search criteria.” This is a very theoretical definition, but if we analyze this, all it says is you have an input variable (the specific search criteria) and based on that variable you should find some other information (particular item of information) you want to use. Examples of relations are “Secretary of,” or “Manager of.”

Using relations in process design allows activities to be assigned in a flexible way. Relations are usually applied to enhance organizational concepts or to describe individual structures. Typical relation names are “Secretary of” or “Department of.” Relations can also be used independent of any organizational structures, for instance, “Part No. of” that returns the part number of a given item.

Relations are defined in the organization database and can be used in multiple processes. If a relation changes you have to change only the relation document in the organization database and the changes will be reflected in runtime immediately.

There are already some standard relations defined in the organization directory, for example “Manager of employee” where the input will be the employee name and the result will be the manager of that specific employee as defined in the department document. These can be used in the Architect while defining your process and activity properties. In addition, you can also define your own relations.

The result of a relation is either a field value or a column value. When you are using a column value, the relation will stop at the first document found. When you are using a field value, it can be either a single value or multiple values that the relation will return. Be aware of the fact that a relation can only relate to one document. Therefore, one to many relations are possible on the field level (multiple value fields) but not on the document level (only one document is selected).

Relations are often used to assign activities to certain persons related to a workflow participant. For example, a travel request should be approved by the manager of the initiator. In the Domino Workflow Architect you can use relations to calculate activity owners, as shown in the example that follows. Once a relation is defined in the organization directory database, it can be used in Domino Workflow formula expressions.

Using relations – an example

This example uses relations to assign a certain activity to a specific person or group based on a product code. The product code will be information the initiator will fill in on the form (in a field). For every product code another

person will be the activity owner of the next activity. If you want to make that kind of relation, you need to make the following changes.

Changes in the organization directory

First, you have to change the person form in the organization directory. We will make the change in the custom subform "PersonInclude." Do the following:

1. Open the organization directory in Domino Designer.
2. Open the subform "Person Include."
3. Add a field to this subform called "ProductCodes" (keywords, text).
4. Save and close the subform.

After adding the field you must create the relation document. To create relations, your name must be in the owner field of the setup document for the organization directory. Make sure you have owner rights in the organization directory and then follow these steps:

1. Go to the Relations view in the organization database.
2. Click the "Create relation document" button.
3. Give the relation a name using the Name button. In our example we used:
Product Manager of
4. Select the database where you want to look up the relation. You can choose among the organization directory, process definition, or any other database specified by either replica ID or path and filename. In this example, we use the organization directory.
5. Give the type of relation. In this example we used "Search within documents."

Note You can either search within a collection of documents or within views. When you search within documents you must specify the selection formula for the documents to look in (see the following step). If you search within views you must specify the view name.

6. Write the selection formula for documents to look for this relation in this example:
**Form = "Person" & @ismember(@Input; ProductCodes) &
@isunavailable(InconsistentOS)**
7. Write the name of the field where the value for the relation is stored. In this example we use:

CurrentNameOS

The field "CurrentNameOS" is where the name of the person (our product manager) is stored.

Note When you specify a view instead, you must specify which column in the view holds the value for your relation (called the result column).

8. Save and close the relation document.

The relation document should look like the following figure (in edit mode).

The screenshot shows the configuration for a relation in Domino Designer. The title bar reads "Product Manager of". The "Name" field is set to "Product Manager of". Under "Database", the "Organization Directory" radio button is selected. Under "Type of Relation", the "Search within documents" radio button is selected. The "Document Selection Formula" is: `Form = "Person" & @ismember{@Input; ProductCodes} & @isunavailable{ InconsistentOS }`. The "Result field" is: `CurrentNameOS`.

The relation is now created in the organization directory. To use it, change your process in the Architect and your form in the application, as described below.

Changes in the application database

You need to add a field on your form where the user can specify a ProductCode. You will use the product code to find the product manager for the product and then make that person the Activity Owner. To add the field, follow these steps:

1. Open your application database in Domino Designer.
2. Open the form used in your process.
3. Add a field to the form called:
ProductCodes
4. Make this field a keywords field with the same options used in the organization ProductCodes field.
5. Save and close this form.

Changes in the Architect

Now go to the Architect and follow these steps:

1. Open the specified process where you want to use the relation.
2. Select the "Basic properties" of the activity where you want to use the relation.
3. Click Select on the Owner tab.

4. To use your relation you must create a formula in the Architect. Do this with the following steps:

- a. Select Formula in the selection screen.
- b. Click the Create formula button.
- c. Give the formula an appropriate name, for instance

Product Manager

a. Specify the content of the formula:

```
@Relation([Product Manager of]; ProductCodes )
```

This formula defines the field that will be used as input for the relation.

Tip You can also select your relation from the library by clicking the Formula button and then the Library button.

5. Close the formula by clicking OK.
6. Close the Basic properties by clicking OK.

Now you must activate the new version of your process to your runtime environment to test your changes. Before you test your process, you should fill in the field, ProductCodes, with the appropriate content on some person documents so there will be results of the evaluation of the relation.

Tip A relation may not be defined for every possible input value. In the example used here, there may be a product code to which no product manager has been assigned. If a relation is being used within a process and cannot be evaluated properly given the context of a specific job, a routing error will occur. You can handle this type of problem within your formula using the @IsError function. An example for an Activity Owner formula may look like this:

```
PM:=@Relation([Product Manager of]; ProductCodes );  
@if(@IsError(PM);@JobProperty([JobOwner]);PM)
```

Job properties

Job properties are formulas that represent information generated during the course of a job, and that is therefore not known during design time. When you define or use job properties, always consider all possible return values to avoid errors at runtime. Using job properties you can get access to the information generated during the execution of a specific job at the moment you need the information and the information is generated.

Some examples of how you might use job properties are the following:

- To assign participants to a specific activity, such as Job Owner, Activity Owner, or Previous Activity Owner.

- To define routing relations based on, for example, “Decision of Activity” or “Number Of Executions Of Activity.”
- To extract information available in binder documents, such as “Job Priority” or “Due Date.”

To create job properties, your name must be in the owner field in the setup document for the organization directory. To create a job property follow these steps:

1. Go to the Job Properties view in the organization database.
2. Click the Create a Job Property document button.
3. Give the Job Property a name using the Name button.

Note There are predefined job properties. Their names are reserved and cannot be used to create new job properties. The names and functions of the predefined job properties are listed in the table below.

4. Enter a formula in the formula field.

In the example in Chapter 6, “Introducing Domino Workflow in an existing application,” the JobProperties were used to get a list of reviewers, select every reviewer personally, and assign them to another activity.

The predefined job properties offered by Domino Workflow are identified in the following table.

<i>Name</i>	<i>Description</i>	<i>Output</i>
@JobProperty([Activity Due Date])	Shows the activity due date.	Time date
@JobProperty([Name of activity]; “Name of the activity”)	Provides the name of the activity owner of a given activity. If the activity was worked on by more than one activity owner, the most recent one is listed.	Text list
@JobProperty([Activity Owner])	Provides the name of the activity owner of the current activity. If the activity was worked on by more than one activity owner, the most recent one is listed.	Text list
@JobProperty([Activity Priority])	Provides the activity priority of the current activity.	Text
@JobProperty([Approval])	This job property is only supported for reasons of upward compatibility. It has the same function as decision. Use decision instead.	Text

continued

<i>Name</i>	<i>Description</i>	<i>Output</i>
@JobProperty([Decision])	Shows the decision that was made by the activity owner at the preceding activity, that requires a decision from the given alternatives. This Job Property should not be used after binders have been joined, because the result in this case is undefined. Use the "Decision of Activity" instead.	Text
@JobProperty([Decision of Activity]; "Name of the activity")	Shows the decision that was made by the activity owner of the specified activity.	Text
@JobProperty([External Initiator])	This Job Property is available if a job was initiated by mail or form-based initiation. It returns the value of the field "ExternalInitiatorOS" of the Cover document.	Text list
@JobProperty([Initiator])	Shows the initiator of a job. The initiator is the person that completed the first activity.	Text list
@JobProperty([Job Due Date])	Shows the job due date.	Time date
@JobProperty([Job Owner])	Shows a list of job owners.	Text list
@JobProperty([Job Priority])	Shows the job priority.	Text
@JobProperty([Number Of Executions Of Activity]; "Name of the activity")	Shows how often an activity was performed during a job.	Number
@JobProperty([Previous Activity Owner])	Shows the activity owner of the previous activity.	Text
@JobProperty([Tasks Done])	Shows the completed tasks of the previous activity. This job property should not be used after joining binders, as the return value then is undefined.	Text list

Resources

In the organization directory navigator there is a "Resources" button, which opens the view "7. Resources". All resources are listed in this view. A resource is a document identifying some aspect of your company's information technology infrastructure. A resource can be either an application database address, a mail address, a client-driven program, or a server-driven program. The four types of resources are discussed below.

Application databases

Application resources are documents that contain mail addresses to application databases. When you are using subprocessing in other databases, the different databases should be registered as resources in the organization database. For more information on subprocessing, see Chapter 7.

Note Both application databases also need to be defined as mail-in databases in the Domino Directory. Each application's resource name must also be entered in the setup document.

An application resource is technically the same as a Mail resource. However, the application resource is dedicated to use for application databases. You are also able to specify in what format the other database should be informed. Here you can specify XML or Notes format.

Mail addresses

The mail addresses defined in the organization directory can be either those of databases or of persons. A mail address resource is nothing more than the mail address of the specified person or the mail-in name of the specific database that can be used for designing processes in Domino Workflow Architect.

If the mail address of a person that is defined in the Domino Workflow Organization Directory is different from its user name (for example, an Internet mail address or a mail address defined in a different domain), you do not need to define a mail resource. Person documents contain a dedicated field to specify a mail address other than user name.

Note An application database is also a mail-in database, but must be specified as an application database. Domino Workflow differentiates between these two sorts of mail addresses because the information is used in different parts.

There are various kinds of databases that can (or must, if you want to use them for your workflow) be specified in the resource view:

- Audit trail database (only when the audit trail is done in a separate database).
- Archive database (only when archive is enabled in a separate database).
- Mail-in databases used in automated activities.

Note Resource mail addresses pointing to some databases must be defined as mail-in databases in the Domino Directory.

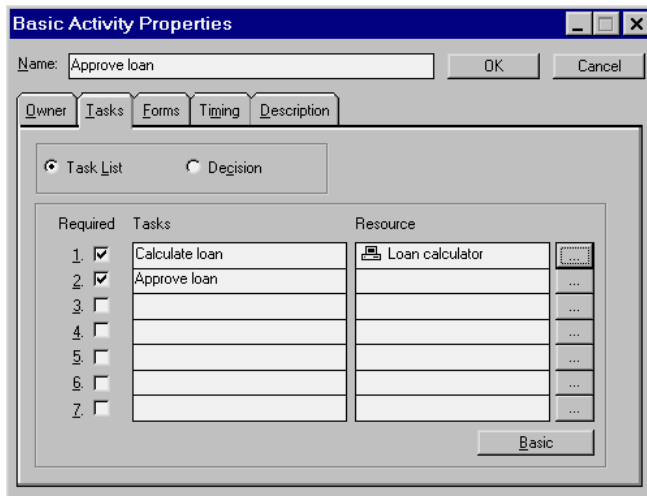
To create a mail address use the button to directly get the databases from the Domino Directory.

Client-driven programs

Client-driven programs are resources that can be launched on any client computer used by Domino Workflow participants. Client-driven programs can be stored in the organization directory as installable files, or can just define a path and a command name for an executable on your hard disk.

If you want to see an example of installable files, install the Domino Workflow viewer. Otherwise, give the path and the filename of the specific program on your hard disk or network.

In the Domino Workflow Architect you can assign tasks to workflow participants and add resources to these tasks. The resources added to a task can invoke either forms or client-driven programs. You can see an example of such a task in the following figure. In this figure a resource program is created that helps the participant to make a certain loan calculation. In the runtime environment, the Domino Workflow user will find a button in the binder document which will launch this program. To see and use this button the document has to be in “Edit” mode.



Clicking this button will launch a program with the parameters as defined in the Architect. It will allow the workflow participant to concentrate on the real task instead of bothering with the technical side of the activity, such as which calculation program to use and with which parameter.

Server-driven programs

Server-driven programs are programs that can be launched from within the Domino server. The path and filename of a server-driven program is stored in the organization directory. Server-driven programs can only be launched in an automated activity. Since the program is launched by the background

agent in the Domino Workflow application, the complete path and filename should be entered in the resource document.

Examples of some scenarios where a server-driven program would be useful are the following:

- If an order is placed during a process, an automated activity can start a program that dials up the pager of the person responsible for ordering the goods, to inform them about the new order.
- In credit applications, actual transactions may be conducted by triggering mainframe procedures.

Alternative organization directories

Instead of using the organization directory of Domino Workflow you can decide to directly use information stored in your current organization directory, whether it is the Domino Directory or some other custom directory that your company may have created. You need the Domino Workflow Developer's Toolkit 2.0 to use another organization directory. The Toolkit comes with examples of how to use Domino Directory and the Lotus Solution Architecture (LSA) organization directory with Domino Workflow.

- **Domino Directory**

The Domino Directory contains all kinds of organization information. All people are stored in this database, and groups can be created to group the people within your organization. Lots of companies do not want another organization directory where all organization data is also stored. Therefore, it is possible to integrate your Domino Workflow environment directly with the Domino Directory.

- **LSA Organization Directory**

The LSA Organization Directory is an asset developed by Lotus Professional Services EMEA. It acts as a front-end to the Domino Directory and allows you to manage Domino Directory groups from an organizational point of view. Besides in many other features, it also allows you to apply surrogation rules to the Domino Directory.

Note Domino Workflow can also be customized to support other Directories. The integration with the Domino Directory and the LSA organization directory are just examples of such a customization.

Integrating an alternative organization directory

You have to make some adjustments to the Architect and to your application database to integrate an alternative organization directory, as follows:

1. Change the mapping files (explained below) in the Domino Workflow Architect directory.
2. Replace some design elements in the application database using the Domino Workflow Developer's Toolkit (For more information on the Toolkit, see Chapter 7).
3. Change your setup document in the application database.
4. Adjust your database settings in the Architect to change the organization database and appoint a resource database.
5. Adjust your process using the new organization directory.

Mapping files

Domino Workflow uses some mapping files for the Architect. The following mapping files can be found in the Architect directory after installing the product.

<i>Mapping file name</i>	<i>Description and use</i>
Architct.ini	This mapping file is used to configure the Architect with user-specific information. This file stores profiles and preferences of the specific user. Also the last processes used are stored here.
dwforgre.ini	This mapping file is used when the Domino Workflow organization directory is used as both organization and resource database.
dwfre.ini	This mapping file is used when an external organization directory is used and the Domino Workflow organization database is used as resource database to store system-specific data.
LSAorg.ini	This mapping file is used when LSA is used as the alternative organization directory.
NABorg.ini	This mapping file is used when the Domino Directory is used as the alternative organization directory.
Workflow.ini	This mapping file is used for all graphical tools (Architect and Viewer). You can find the settings like language and style sheets in this file.

When you want to use an alternative organization directory, two changes must be made in order to have access to this directory from the Architect.

- First, you must specify a resource database in the database profile in Domino Workflow Architect.

A resource database is the database where information is stored which cannot be stored in the alternative organization directory. Your alternative organization directory probably will not support all objects defined in the Domino Workflow organization database. In this case, the Domino Workflow organization database has to be used additionally as a resource database.

When you use the Domino Workflow organization database only as a resource, many of its design elements won't be required any more. You can hide or modify these design elements, as desired. For example, the navigator can be changed and the buttons for persons, departments, roles, and workgroups can be deleted.

- Second, you must specify how information from the alternative organization directory should be interpreted by the Architect. This can be done in a mapping file. You can add your own views that can be read by the Architect. Depending on the type of alternative organization directory, you can add and change some elements of the specified file.

Specifying an alternative organization directory and a resource database

In order to use the Domino Workflow Organization Directory only for storing Domino Workflow specific information and not the organizational structure, you have to make the following mapping changes in the Domino Workflow Architect directory:

1. Adjust Architect.ini, add the following row in the "[Configuration]" section of the file:

AllowResourceDatabase=yes

Note This can be done by double-clicking Architect.ini. A text editor will be started and you can type in the text. After changing the file, save and close the mapping file. The change will be effective when the Domino Workflow Architect is reopened and will allow you to specify a resource database.

2. Make the following mapping file changes:
 - a. Add the dwforgre.ini to use the Domino Workflow organization database as resource database.
 - b. Add the LSAOrg.ini if you want to use the LSA organization directory or the NABOrg.ini if you want to use the Domino Directory as alternative organization directory.

Note This depends on what kind of organization directory is used in the Domino Workflow application.

Tip In the Domino Workflow Developer's Toolkit you will find more information about the mapping files and changes you can make to them.

Adding your own OrgaTypes to the Architect

You can create your own organization types such as Project Teams or Positions based on folders and views in the organization database. These organization types must be added to the mapping file before you can use them in the Domino Workflow Architect. Your own OrgaType will be very useful in large organizations to group your Domino Directory groups in more functional groups. This way you do not have to scroll through all the groups to get the one you are looking for.

Note Whether or not you are using an alternative organization directory, you are always able to add your own OrgaTypes to the Architect. You only need to change the mapping file used for the organization directory. Mapping file types are discussed above.

The following example demonstrates how to create your own OrgaTypes:

1. Create a folder in the Domino Directory called "Departments."
2. Drag the groups which are filled with department members into the folder.
3. Go to the Architect directory.
4. Open the naborg.ini mapping file in a text editor.

Add your own type to the "SupportedTypes" in the "[General]" section (on top of the mapping file):

```
SupportedTypes=Person, Workgroup, Departments
```

Add the following lines at the end of the mapping file (these are the lines that refer to the specific folder):

```
[Departments]

Selection=View#Departments

SystemName=Field#Listname#TextList#0

DisplayName=Formula#Listname + " , " + Type

TypeBitmap=File#LSA.bmp
```

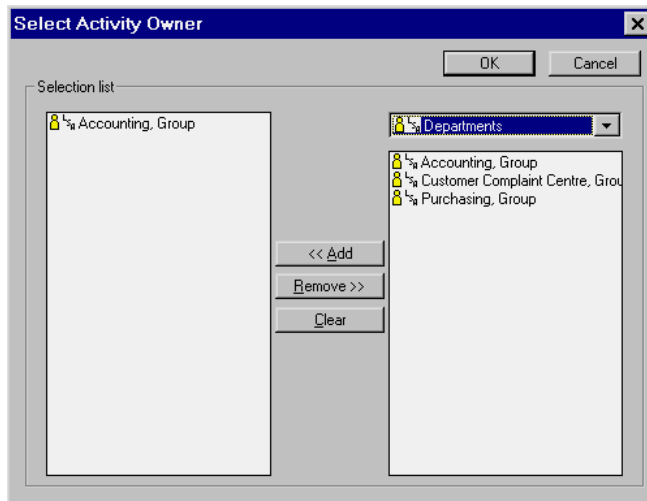
In the line starting with `selection=` we specify departments as view or folder containing the relevant information. The next line specifies the name of the field containing the specific group you want to use for the process design. In the line starting with `DisplayName=` you should specify the name that will be used for display in the Architect. In this example we used a formula to display the name.

In the last line the LSA.bmp is specified in order to illustrate that a bitmap can be assigned as a symbol to identify your own organization types in the Architect. The LSA.bmp is available to everybody with a standard Domino Workflow installation, so that is why we used this file in this example. You are free to change it to any kind of bitmap file, as long as the file is stored in the Architect directory.

Note All bitmap files used in the Business Object Library of the Domino Workflow Architect are 22x12 pixels.

5. Save and close the mapping file.
6. Start the Domino Workflow Architect and you will now be able to select people from the type “Departments” as shown in the figure below.

Note If your Domino Workflow Architect was running during the changes, you have to restart it in order to activate your changes.



Replace design elements based on the Toolkit

You must replace some design elements in the application database to use an alternative organization directory. Domino Workflow needs some design elements to interpret the information from the alternative organization directory. When integrating with the Domino Directory, the following design elements must be replaced:

- Script library: “OS Application Events” containing events which you can customize to your specific needs and demands. Compared to the standard “OS Application Events” library, this version contains several additional Events dealing with Organizational issues. In this library is the LotusScript code used to create the integration. Script library: “OS OAM Events” contains support for the additional organization access events.
- Sub form: “OS Domino Workflow CustomApplicationSetup” contains an extension to define your Domino Directory in your Application Setup Documents.

The design elements can be found in the Domino Workflow Developer’s Toolkit which we described in Chapter 7. The elements are also documented in the Toolkit.

Change your Setup document

The Domino Workflow organization database and the Domino Workflow process definition database, used in conjunction with the application database, must be entered in the application setup document in the Domino Workflow application database.

After replacing standard subform “OS Domino Workflow CustomApplicationSetup” with the new “OS Domino Workflow CustomApplicationSetup” subform from the toolkit, the following section appears (in your application setup document), where you can define the Domino Directory settings:

The screenshot shows a subform titled "textCustom design" with a section for "Domino Directory settings". It contains two fields: "Server" and "Path". The "Path" field is currently set to "names.nsf". Below the fields is a label: "textReplicaID or Path of the Organization Directory Database (Path if local)".

The server field is set automatically (Read only). You supply only the path to the Domino Directory.

Note The Domino Directory must be located on the same server as the Domino Workflow application database and the Domino Workflow organization database.

In the organization database section you can define the settings of the resource database. Because the Domino Workflow organization database is used as a resource database, you can retain the settings provided.

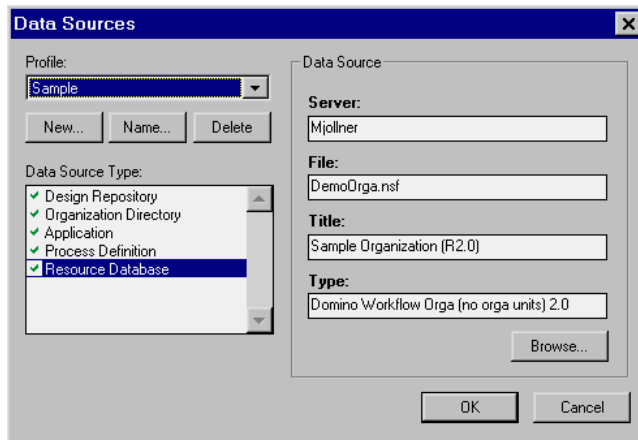
Change your data sources in the Architect

When you start the Architect you must choose the right databases for your process. To do this follow these steps:

1. Start the Domino Workflow Architect
2. Choose File - Open Databases
3. Select the organization directory and change the selected database by referring to Domino Directory or your LSA organization directory database.

Note The database type will be generated automatically.

4. Select the Resource Database and select your Domino Workflow organization database as shown in the figure below.



Note Here the database type will be generated automatically.

5. Click OK and the correct databases are selected. You are now able to use the alternative organization directory in your process design.

Change and reactivate your processes

You have to bear in mind, that once the settings in the Application database are changed, all organizational units will be evaluated against the directory that you have defined. Therefore, you may need to change your process definition and reactivate your processes and then update existing jobs using the reroute feature.

Integrate with other organization directories

As mentioned earlier in this chapter, Domino Workflow does not provide out-of-the-box integration with other organization directories besides Domino Directory and LSA organization directory. If you want to integrate with other organization directories, such as LDAP, Active Directory, and so forth, there are some things you should be aware of.

1. In the engine, you have to add your own code to the following events defined in the “OS Application Events” Script Library:
 - **OnEvaluateParticipants:** This event is called whenever the engine evaluates which users are assigned to an organizational unit.
 - **OnEvaluateEMailAddress:** This event is called whenever the engine needs the mail address of a user or group.
 - **OnEvaluateOrgaType:** This event is called whenever the engine needs to determine the type of an organizational unit.
 - **OnNewOAM :** This event is called whenever the participant evaluation is initialized. Use this event to set global variables such as a reference to your directory. You may define the path to this directory by adding fields to the custom subform used in the Application Setup document.

For more details and examples, refer to the Domino Workflow Developer’s Toolkit.

Note If you are using a customized Domino Directory, you may be able to use the code provided by the Toolkit for Domino Directory integration. You can analyze the code and add your own extensions.

2. You should write your own mapping file so the Architect will know where to look for the organization directory data. This is documented in the Toolkit.
3. The Architect can read only from Domino databases. If you want to use an organization directory other than Domino databases, you currently have to convert your organization directory data into a Domino format.

Functions to point the Architect against directory sources other than Domino databases are being developed. Check the Domino Workflow Web site for updated information about the availability of this functionality:

<http://www.lotus.com/workflow>

Summary

In this chapter we discussed two main topics:

1. Advanced use of the Domino Workflow organization directory.
2. Integration with another Domino organization directory.

Advanced use of the organization database will give you opportunities to build a maintainable application. Relations and Job properties can be very useful. In addition, you are also able to define resources such as application databases, mail addresses, server-based programs, or client-based programs. These resources offer you the possibility of linking your application to other parts of your information technology infrastructure.

When you use Domino, a lot of organizational data will be registered in the Domino Directory. Therefore, many organizations do not want another organization database such as the Domino Workflow organization directory. Using the Domino Workflow Developer's Toolkit, you are able to integrate your Domino Workflow application with your current organization directory.

Note The Domino Workflow Developers Toolkit provides design elements for integration with the Domino Directory or the LSA organization directory. It also provides hooks to implement your own integration with other organization directories.

To make the integration available, you need to make changes in the setup of your database and the setup of your process. Also, some mapping files need to be changed. You can also create your own OrgaTypes and access them by defining views in your mapping files. The integration of an alternative organization directory can be implemented within a reasonable timeframe.

Chapter 9

Distributed processing

In this chapter we show several examples of distributed processing in a Domino Workflow environment. We divided them into two categories:

- Multiple Domino Workflow application databases
- Multiple replicas of Domino Workflow databases

Distributed processing represents an important way to tune your Domino Workflow application in size, performance, and structure.

Multiple databases

The Domino Workflow solution can consist of multiple Domino Workflow application databases. They can share the same server and process definition database or they can be on different servers and have separate process definition databases.

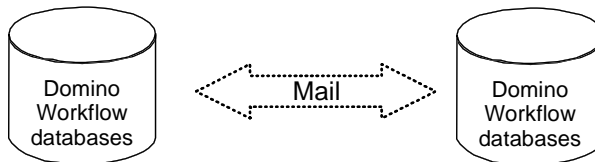
Process subsets

In the application setup document, you can specify which processes can be started in that particular database. In this way, the complete set of processes can be divided between several databases. For example, a contract management system might have a Domino Workflow application database for the different contract related workflows and another Domino Workflow application database for customer related workflows. Both will be part of a single system and share the same process definition database and organization database, but have their own set of processes. These Domino Workflow application databases would be placed on the same server. The advantages and disadvantages of this architecture are summarized in the following table.

<i>Advantages</i>	<i>Disadvantages</i>
A single process definition and design repository database.	Maintenance of multiple Domino Workflow application databases.
Some processes can be started in more than one database.	

Subprocesses

Subprocesses are generally used because these subprocesses are owned by different parts on an organization (or even different organizations) than the host process. A credit approval may, for example, be usually handled in a branch office directly, however, if the credit volume is high, an additional approval process has to be started in the central bank office. The target database for the subprocess is reached through mail, so it is even possible that the subprocess will run on a different server than where the host process runs. The advantages and disadvantages of this configuration are summarized in the next table.



Advantages

Cross department/organization processes possible.

Possible to use different servers for host and subprocesses.

The application in the subprocess runs can use a different process definition database than the host process.

Disadvantages

Maintenance of multiple Domino Workflow application databases is required.

Multiple replicas

The Domino Workflow databases can be replicated to other servers or they can be replicated locally.

Caution The Domino Workflow agents should only be scheduled on a single server. The chance of replication conflicts sharply increases if the Domino Workflow agents are scheduled on more than one server.

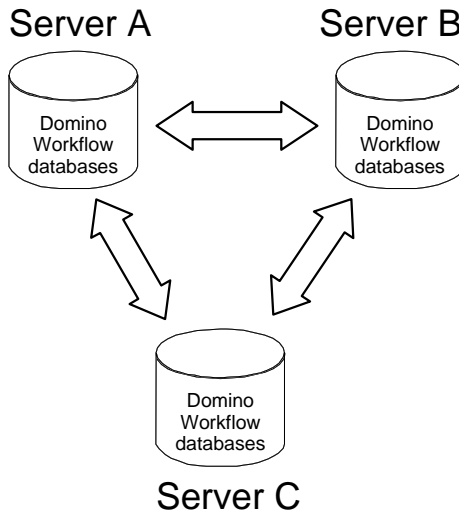
All the databases which are needed in a production environment (Application, Process Definition, and Organization) should be available on the same machine, either locally or on the server. You will not be able to work with the application database if all databases not are available.

Domino Workflow normally operates in such a way that either the server or the activity owner acts on a document, but not both at the same time. This reduces the chance of replication conflicts.

If the configuration you choose has multiple servers, you should be aware that because the server on which the Domino Workflow agents are scheduled is different from the server which (most) end users will access, they might get notifications for jobs that haven't arrived yet on their server. You cannot completely prevent this from happening, but if replication is scheduled more often than the agents are scheduled, the chances of this happening are reduced.

Server cluster

The server on which the Domino Workflow databases reside can be part of a cluster. This can improve the performance and especially the availability of the databases. Apart from the general notes at the beginning of this section, at this time, there is no known downside to using Domino Workflow in a server cluster.



<i>Advantages</i>	<i>Disadvantages</i>
Higher server availability.	Higher infrastructure cost.
One of the servers can be used as an Agent server (see below).	

Local replicas - mobile users

If you allow your end users to work on a local replica, it is important that all the Domino Workflow databases are replicated. The Notes client will look for the process definition database and the organization database in the same location as the application database. Working on a local replica increases the chance of replication conflicts (especially if there is more than one potential

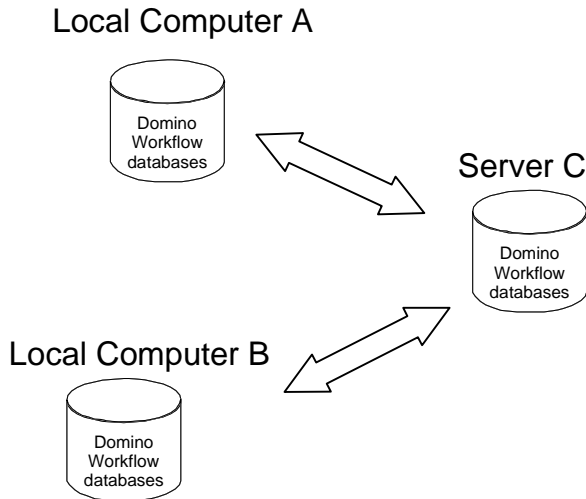
activity owner defined or when a team activity is defined). To reduce the risk of replication conflicts, we recommend that you:

- Restrict the number of potential activity owners to a minimum.
- Only claim jobs on the server replica (not in the replica copy on your local machine).

Note See the section on selective replication for ways to keep your database size small.

Protecting documents

Another way to prevent replication conflicts, and avoid a situation where several people unknowingly are working locally on the same documents, is to have the team members lock documents in the binder they want to work with offline by using the action “Save & Write Protect,” which can be found on all documents in a binder. The protection can be switched off via the “Release Protection” action.



Advantages

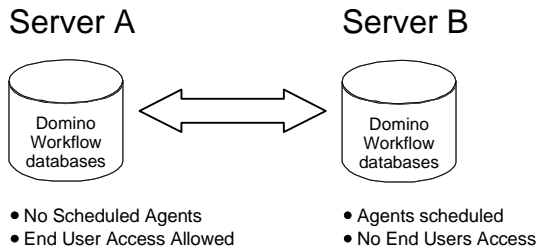
Better performance than over a network.
Users have the ability to work offline.

Disadvantages

Higher chances of replication conflict.
Dependent on replication schedule of end user.
Data not up to date on server and on local drives.
Increased disk usage on the client side.

Using an agent server

If the processes you defined contain many automated activities, which take a lot of processing power, you can add an extra server on which you schedule all Domino Workflow agents. This server can be part of a server cluster or connected to the other server via a normal replication schedule. The end users should not have access to this server.



Note If performance is an issue it is also very important to keep the databases as small as possible, for example through archiving completed jobs, because one of the more demanding tasks of the Domino server is keeping the indexes of all the views up to date.

Advantages

End users don't notice the workload of agents.

Can be combined with a server cluster.

Disadvantages

Notifications can get sent too early.

Higher infrastructure cost.

Location replicas

People who access the Domino Workflow server from distant locations may experience longer response times than users sitting on the same LAN as the server. This is often due to network latency. Network latency is the time it takes for the client to get an answer on a question. The Notes client is sensitive to network latency because it generally has a lot of 'small' transactions with the server for each user action. There are basically two ways to solve this:

1. Redesign the application to make it less sensitive to long network distances.
2. Locate the server closer to the users.

For advice on designing for performance, see the redbook *Performance Considerations for Domino Applications*, SG24-5602, Lotus part number CT7V6NA. To minimize the distance between servers and users, consider adding new servers that are close to groups of users. All the servers need to be connected through a replication schedule.

<i>Advantages</i>	<i>Disadvantages</i>
Better performance in case the network distance for users becomes shorter.	Higher infrastructure cost. Notifications can get sent too early.

Selective replication

Selective replication means that only a subset of all the documents is replicated. Selective replication can be between Domino servers or between a Notes client and a Domino server.

Tip If you can define a few typical user groups, you can define a view for each that will contain all the documents each particular group needs. In this way the replication formula is part of the design instead of a replication setting. Include only one column with, for example, the document number (@DocNumber won't take up any view index space) to keep the view as small as possible.

Server to server

The most difficult part of selective replication is to avoid ending up with an inconsistent subset of all the data. For example:

- Binders without a Cover document
- Binders without the Main document

One way to avoid this situation is to formulate the replication in a negative way. So, you would formulate what you **don't** want to have in this replica. For example to leave some process out of a certain replica you would use:

```
Select !(ProcessIdOS = "SomeProcessA1")
```

or, to leave out the audit trail documents:

```
Select !(Form = "(OS Audit Form)" : "(OS Audit Merge))"
```

Note Be careful, though, when comparing list values with scalar values. For example the formula:

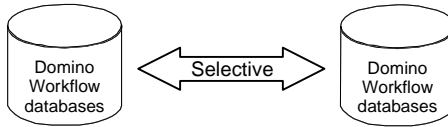
```
!("a" : "b" : "c" = "a")
```

is **False**, whereas the following formula, which looks similar, is **True**

```
"a" : "b" : "c" != "a"
```

Client to server

Client to server replication works in the same way, except that the administrators of the databases have no control over the formula. It is up to the end user to specify the replication formulas, and it can be difficult to track down erroneous behavior of the application at the client side if selective replication is used.



Advantages

Smaller Domino Workflow databases.
People get only the data they need.
Views can be used to preselect data sets.

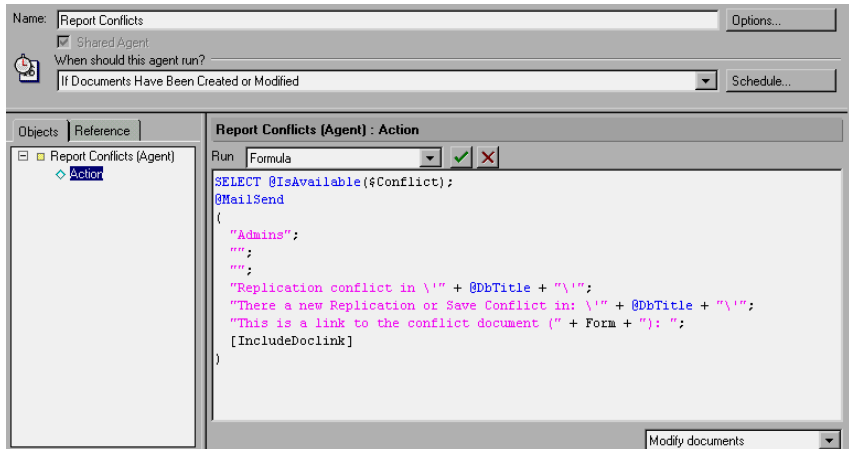
Disadvantages

The replication formulas are difficult to maintain.
Increased chance of replication conflicts.
Additional administrator tasks.
Knowledge about Domino Workflow fields is necessary.
Client to server selective replication is difficult to manage.

Handling replication conflicts

It is important that all replication conflicts are repaired as soon as possible, because binders will fail to route if the binder contains a replication conflict document.

To be able to get an early alert that a replication conflict has occurred, you can schedule an agent defined as in the following figure.



This agent will send a memo to the members of the group Admins if there are replication or save conflict documents.

Summary

In this chapter we discussed two kinds of distributed processing in a Domino Workflow environment. The workflow load can be distributed across different Application databases and servers by using several Application databases or by using Replication. Multiple databases can be created and limited only to support dedicated processes. Workflows can span multiple Application databases based on the subprocess feature. This also allows different parts of an organization to be responsible for specific subprocesses.

Alternatively, replicas of Domino Workflow databases can be used to overcome network limitations as well as server load restrictions.

Chapter 10 Integration

In this chapter we discuss two integration possibilities for Domino Workflow:

- Integration with Domino.Doc
- Integration with other workflow systems via XML

As we mentioned in Chapter 2, Domino Workflow builds on top of Domino and can utilize all services offered by Domino. Therefore, there also are integration possibilities with products like DECS, MQSeries, and so on. Integrating these products with Domino Workflow is the same as their general integration with Domino applications. To learn more about the integration options check out the Redbooks about Domino enterprise integration listed in the “Related publications” section in the back of the book.

Domino.Doc

Domino.Doc is a document management product that supports distributed document management and handles the complete life cycle of documents: authoring, review, approval, distribution, and archiving. Built on top of Lotus Notes and Domino, it leverages the power of Domino to provide replication, scalability, flexibility, and security services.

Five main techniques can be used to integrate Domino.Doc and Domino Workflow. All necessary code to implement any of these scenarios is shipped with the product.

- Initiating Domino Workflow processes from Domino.Doc
- Archiving Domino Workflow Binders in Domino.Doc
- Automatic search in Domino.Doc for specific documents in an automated activity in Domino Workflow
- Automatic check in/check out of documents in Domino.Doc
- Linking between Domino.Doc and Domino Workflow documents

For more information about integration, consult the product documentation, particularly the *Domino.Doc and Domino Workflow Integration Guide* that is delivered together with Domino.Doc as a Domino database.

Access profiles

Some of the building blocks for integration with Domino.Doc are based on Automated Activity agents that communicate with Domino.Doc using the Domino.Doc OLE API. As the API requires the agents to authenticate against Domino.Doc, a username and password has to be provided. As this is sensitive security information, Domino Workflow stores these items in reader field protected documents, the Domino.Doc Access Profiles. Only users with appropriate access rights — and the agents themselves running on server access rights — are able to read (and change) password information.

Domino.Doc Access Profiles can be defined in the Application Setupdocument within the Domino.Doc Integration Section. An example of an access profile is shown in the following figure.

Domino.Doc Access Profile	
What is the name that identifies this profile?	Default
What is the access level for this Domino.Doc account?	<input type="radio"/> high security - only servers and database managers have access <input checked="" type="radio"/> low security - all database users have access <input type="radio"/> custom settings
Read access as computed from the access level settings:	[Process Reader]
What login type shall be used?	<input type="radio"/> via notes <input checked="" type="radio"/> via http
What is the URL of the Domino.Doc library	http://9.95.34.9/8825687B006E8C4F
What is the name of the webuser account?	Admin Freja/Heimdal
What is the password for this login?	redbook

Initiating workflows

You can offer a Domino.Doc user the functionality to initiate Domino Workflow jobs from their Domino.Doc application. You can use event-based initiation or interactive initiation to do this.

Event-based initiation

In event-based initiation, an event that occurs within Domino.Doc initiates a new workflow. Examples of such events are the check in of a new document or the moving of a document from one binder to another. Initiation will start automatically, in the background, without any user interaction. The first activity in the process has to be claimed by one of the potential activity owners of the first activity.

You can enable this feature in the Domino.Doc Document Type settings.

Note The technique used is Form-based initiation.

Interactive initiation

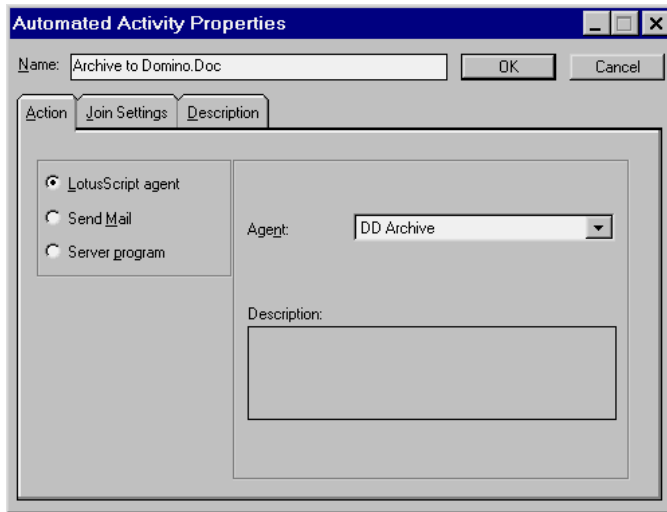
In the “workflow” tab the user can select a process from a list of available processes. The job will be directly initiated from the Domino.Doc client (based on the “New Job” functionality of Domino Workflow). The user will be the activity owner of the first activity when he enters the Domino Workflow application.

The Domino.Doc Library Administrator enables this feature for a given Domino.Doc Document Type. The administrator can also elect to limit a document type to use dedicated processes only, or define which processes are available.

Archiving

Instead of the standard archive feature of Domino Workflow, you can use the Archive to Domino.Doc feature. An automated activity agent is provided in your application database. All binder documents that are routed to an automated activity based on this agent are automatically archived into a Domino.Doc file cabinet. The workflow documents are recreated within a Domino.doc file cabinet, and the original documents are removed from the Workflow application database.

To enable this feature you should create an access profile in the Domino Workflow setup and add an automated activity to your process, as shown in the following figure. You also have to set up a Domino File cabinet to be used as a Domino Workflow Archive. Refer to the Domino.Doc Domino Workflow Integration Guide for more information. This guide is part of the Domino.Doc documentation.



Automated activity search

Domino Workflow provides an automated activity agent which will search Domino.Doc for specific documents. You can assign this automated activity agent to an automated activity to search the Domino.Doc Document Library for specific documents. For instance, you can search based on a customer number for other documents related to this customer. To enable this in your application your workflow binder has to contain a document that contains the following fields that are used as parameters for the agent.

<i>Fieldname</i>	<i>Value Range</i>	<i>Functionality</i>
DDSearchExecute	"Yes", fixed value.	This field acts like a flag for a search request. The DD Search agent skips through the binder and stops with the first document that contains this field with the value set to "Yes".
DDSearchCriterion	Text. Must be a valid full text search criterion.	This field contains the query that will be executed by the DD Search Agent. You can compute the query by using a computed field. This allows you to make the query dependent on a field value, for example: FIELD Customer_Number = \ "" + Customer_Number + \ ""
DDSearchMaxDocs	Number.	This field limits the number of hits returned on the query. Set to 0 to return all matching documents.

Tip The fields can be copied from the "(OS DD Search Setup)" subform contained in the Workflow application database.

Automatic check in/check out

A set of automated activities agents are available in the application database template to check out or check in Domino.Doc documents and retrieve their associated files. These agents all follow the same pattern. First they search the workflow binder for a Domino.Doc Document Reference. This reference is automatically added, when a workflow process was initiated within a Domino.Doc document. However, you can also add your own code to create a reference to a Domino.Doc document before using any of the automatic check in/check out agents. Then the agent authenticates against Domino.Doc using the OLE API of Domino.Doc. The login is done based on the Access profile defined in the application setup. Finally the document is located using the Domino.Doc Document reference and then processed according to the required action. The agents can be used in the automated activities in your process. The following agents are provided for automated check in and check out activities:

1. DD Check Out and Attach

This agent checks out a document from Domino.Doc and attaches the file to a workflow document. The document is checked out under the name of the user specified in the access profile.

Note A Domino.Doc document can only be checked out once, therefore it can only be handled by one workflow at any given time.

2. DD Check In as Version

This agent checks in a copy of the attachment that was added by a “DD Check out and Attach” agent. A new version is created in Domino.Doc. After creating a new version, the document is checked out again. As a result, the file is still owned by the workflow job and the attachment is still within one of the workflow binder documents, but the Domino.Doc document Library contains the new version of the document.

3. DD Check In as Draft

This agent works the same way as the “DD Check In as Version” agent. However, the attachment is checked in as a new draft instead of being checked in as a new version.

4. DD Check In as Version and Detach

This agent checks in a copy of the attachment that was added by a “DD Check Out and Attach” agent. A new version is created in Domino.Doc and the attachment is removed from the Domino Workflow binder. The file is now owned by Domino.Doc again.

5. DD Check In as Draft and Detach

This agent works the same way as the “DD Check In as Version and Detach” agent. However, the attachment is checked in as a new draft instead of being checked in as a new version.

Note Automated activities cannot print into the server log. If an error occurs the job owner is informed via e-mail.

Note If the binder does not contain a reference, the automated activity is skipped.

Note If a document cannot be checked out, an error is reported, but the activity will be repeated until it is finally successful.

Linking between Domino.Doc and Domino Workflow documents

In order to allow users to view a Domino.Doc document from within a workflow binder or to view workflows related to a Domino.Doc document, specific UI building blocks are available.

A Domino.Doc document type can be workflow enabled by setting the Document Type accordingly. After this enablement, all documents created with this type will have an additional tab titled “workflow.” This tab contains all the functionality you need to view related workflows or to initiate new ones (if that is allowed for this document type).

If a workflow has been initiated from Domino.Doc a reference to the Domino.Doc document is added to the workflow document. You can use the subform “(OS DD Connector)” to display the link to a user. It contains all the functionality you need to view a file attached to a Domino.Doc document or to open its profile document. All the functionality within this subform is based on agents shipped with open code. You can easily change the design of this subform or add elements to the forms that you have designed for your process.

Restrictions

If hosting the solution on non-Windows NT servers there is a limitation:

- The automated activities for Archive, Search, and Check in/Check out from Domino Workflow are based on the OLE API of Domino.Doc. Because automated activity agents run on the server, this API has to be available for the LotusScript environment of this server. Therefore, non-Windows NT machines cannot run the automated activities. However, initiation of workflows from within Domino.Doc and the UI integration also work on non-NT servers.
- If Domino.Doc and Domino Workflow are installed on different servers there are some limitations. Please review the Domino.Doc Domino Workflow guide for more information.

XML

XML defined

XML, Extensible Markup Language, is the language for documents containing structured information. Structured information contains both content (words, pictures, and so forth) and some indication of what role that content plays (for example, a process name of a certain job instance).

A markup language is a mechanism to identify structures in a document. The XML specification defines a standard way to add markup to documents.

XML is not a single, predefined markup language. It is a metalanguage (a language for describing other languages) which lets you design your own markup. (A predefined markup language like HTML defines a way to describe information in one specific class of documents. XML lets you define your own customized markup languages for different classes of document.) It can do this because it's written in SGML, the international standard metalanguage for markup.

Domino Workflow and XML

Domino Workflow subprocesses can be used to reduce the complexity of processes and the reuse of parts of a process in other processes. Supplementary subprocesses can run in other Domino Workflow applications.

The Domino Workflow XML interface enhances the subprocess feature and enables connectivity to other workflow management systems. You can use not only Domino Workflow processes as subprocesses, but also processes from other workflow management systems, by implementing a special interface. XML is used to interchange documents and process properties between different workflow management systems.

If a workflow management system implements this interface, Domino Workflow is able to start processes and receive the results of processes from this external workflow management system. Conversely, the external workflow management system can start Domino Workflow processes and receive the results.

To reduce the complexity of the interface, and because of features excluded from Domino and XML, there are some restrictions:

- It cannot be warranted that the same interface will be provided in future versions of Domino Workflow. As there is standardization work in this area, the interface definition is likely to be changed to conform with these standards.

- Only simple data types (String, Integer, Date, and lists) can be interchanged. Notes RichText items, and attachments in documents, are not supported.
- The information from all documents to be interchanged is sent in one e-mail message. If you use Lotus Notes/Domino 4.5 or 4.6, this e-mail must be less than 32K.

Configuring Domino Workflow

Before you can use the XML subprocess interface you must configure Domino Workflow.

Configuring the Domino Workflow Engine

1. Create an Application Resource in the Domino Workflow Organization Database to define the e-mail address of the Domino Workflow Application Database. The Application Resource must have the type "XML".

Note The name of the Application Resource must be the same as the e-mail address.

2. Choose this resource in the "Application" section of the Application Setup document in the Domino Workflow Application Database.

Using external processes in the Domino Workflow Architect

You can use external subprocesses in the Domino Workflow Architect by designing a (normal) Process Link. The only difference between including a Domino Workflow subprocess and an external subprocess is the specification of the target application.

1. Create an Application Resource in the Domino Workflow Organization Database to define the e-mail address of the target application. The Application Resource must have the type "XML".
2. Choose this resource as the target application in the dialog "Process Link Properties" (application tab) in the Domino Workflow Architect.

Communication protocol

Domino Workflow uses a very simple communication protocol between applications. The communication protocol defines which commands are sent by e-mail between distributed workflow management systems.

There are only four different commands:

- JobSubmit: starting a subprocess
- JobRemit: receiving the results of a subprocess
- ACK: positive acknowledgment

- NAK: negative acknowledgment

Starting a subprocess

The workflow management system where the host process is running sends a JobSubmit command to the target workflow management system. Then the target application checks the command. If the target application can start the process successfully, it will send back an "ACK" command. Otherwise it will return a "NAK" command.

Receiving the results of a subprocess

After the subprocess has been completed, the workflow management system in which the process is running sends a "JobRemit" command to the host application. If the command is correct, the host application will send back an "ACK" command. Otherwise it will return a "NAK" command.

More information

For more information on XML, see the Web site at:

www.xml.com

For more technical information and examples, refer to the Domino Workflow 2.0 Developer's Toolkit. See Chapter 7 for instructions on how to get the toolkit.

Summary

In this chapter we discussed how Domino Workflow integrates with Domino.Doc.

We also discussed the concept of integrating with other workflow systems via the XML interface.

Chapter 11

Running Domino Workflow applications

In this chapter we introduce a few concepts related to running your Domino Workflow application. These topics are less important than the design of your processes with the Domino Workflow Architect, but they can be crucial for achieving overall success in introducing Domino Workflow applications in your organization.

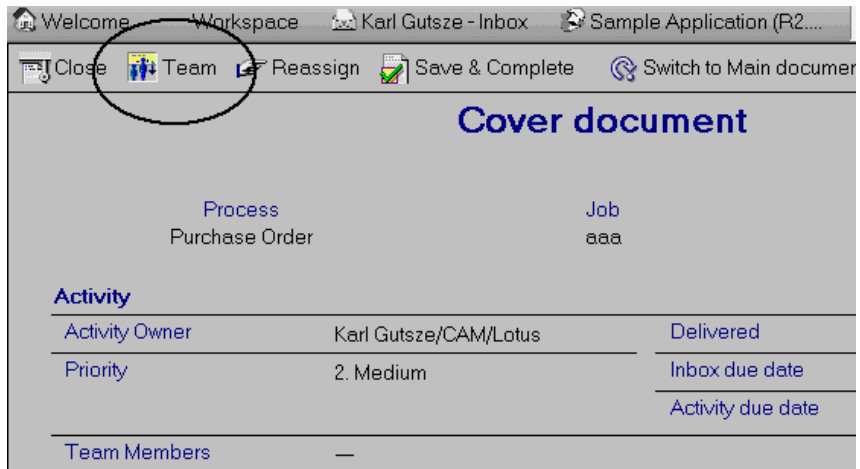
We want to show you a few features that can be useful for running your jobs, as well as some important issues that you should consider when designing your workflow environment. We won't discuss basics about claiming an activity or reserving a document, as they are not only well-documented elsewhere, but are also self-explaining.

Make changes at run time

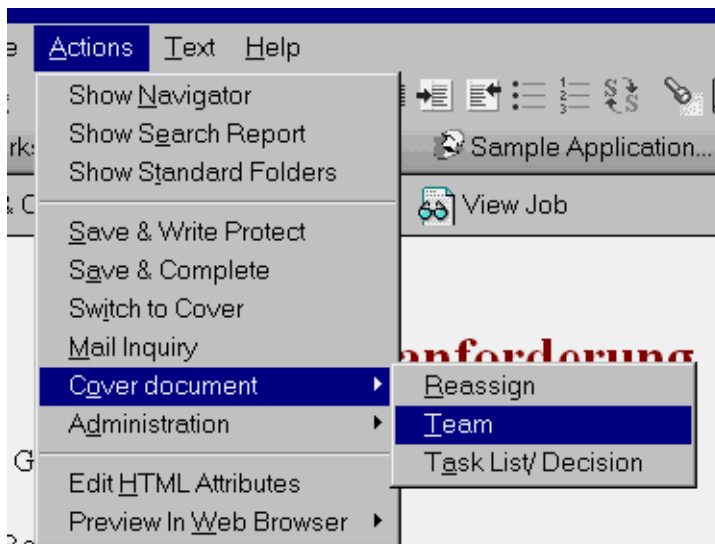
In addition to process design in the Architect prior to deploying the process, you can also make some changes to the process at the time of the job execution. All the changes that you make at run time affect only the particular job, and don't have any influence on other jobs or the process definition.

Change the team working on an activity

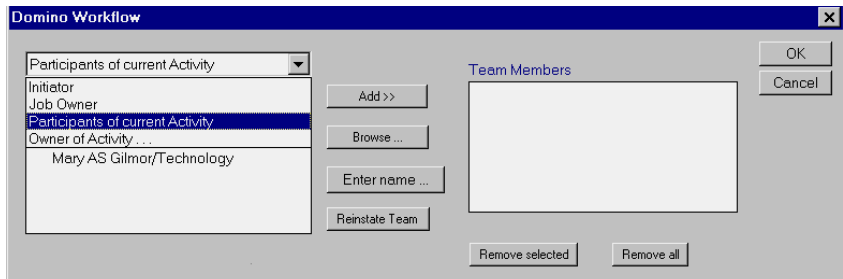
An activity owner has the ability to change the team acting on the activity for which he is responsible. The basic team will be a group of persons, a workgroup, role, or department in the organization database. This team is assigned to the team activity in the Architect at the process level. In some situations it might be necessary to change the team. If this is a structural change you can just change the specific document in the organization database. If this is an incidental change, the activity owner can overrule the static setting. The activity owner must open the cover document in Edit mode and select the "Team" option shown in the following figures.



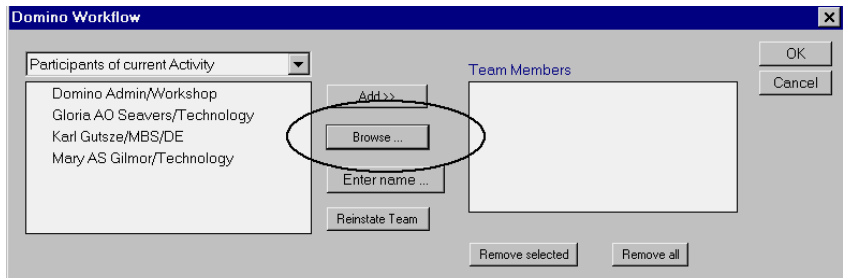
You can also access this action from any binder document that is open in Edit mode. Do this by choosing Action - Cover document - Team.



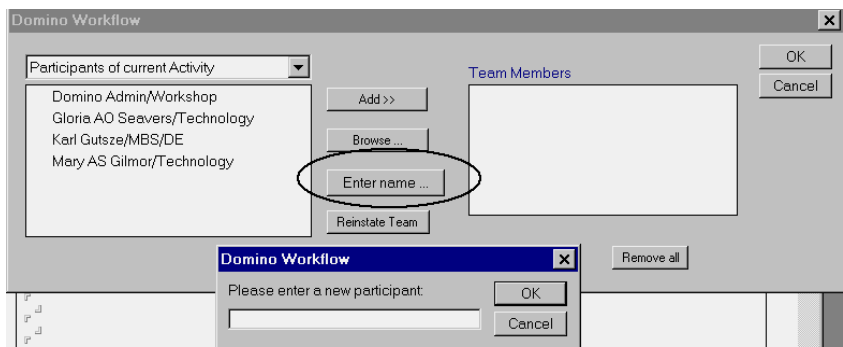
After selecting the “Team” option, you are presented with a dialog box allowing you to change the predefined team (as shown in the next figure). You can add new members to the team, as well as remove existing ones. These changes are valid for one specific activity in one specific job.



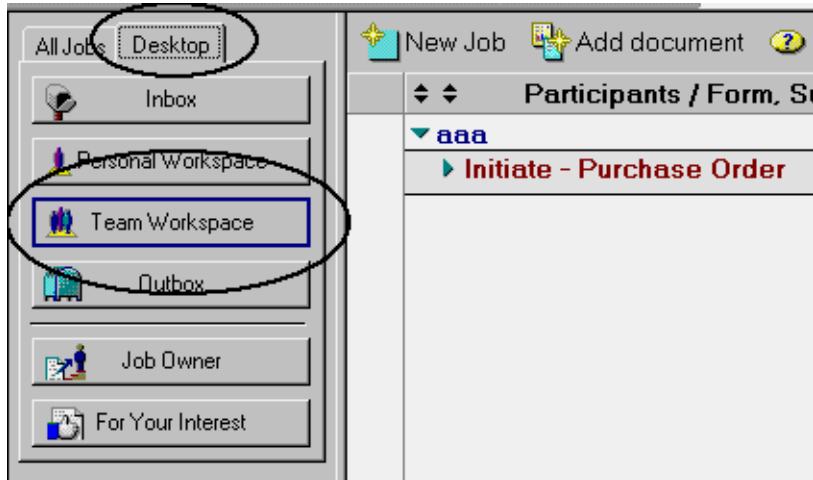
You can choose the team members from your job context (initiator, job owner, potential activity owners of the current activity or an owner of one of the previous activities), or from the context of your whole organization as defined in the organization directory.



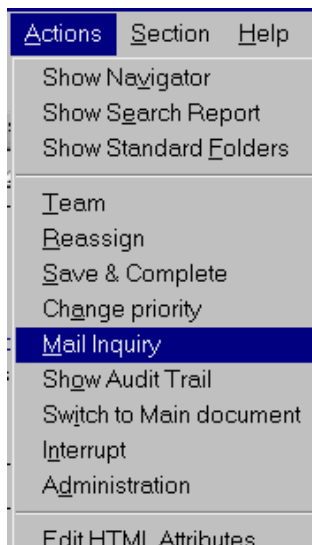
As your organization directory grows bigger, browsing through it might not be the most efficient way to locate people. Therefore, it's possible to choose a person directly by typing their name using the appropriate interface. The new team member has to be included in your organization directory.



By assigning new team members you give them rights to edit workflow-related documents and apply changes to the binder. However, those people don't necessarily know about their new rights. If you use the team feature often, they may be checking the appropriate view.



However, if that is not the case, you should send them some kind of notification. You can do this using the Mail Inquiry feature. You can find this feature in the same document as the Team feature (Cover document). This option is only available when the document is open in Edit mode.



It presents you with a screen in which you can choose people from your organization directory that should be notified, and allows you to send them some text and a link to the specific binder documents. In this way you can notify the team members without sending mail from your own mailbox. A copy of the mail inquiry becomes a binder document. This allows you to keep track of all such correspondence.



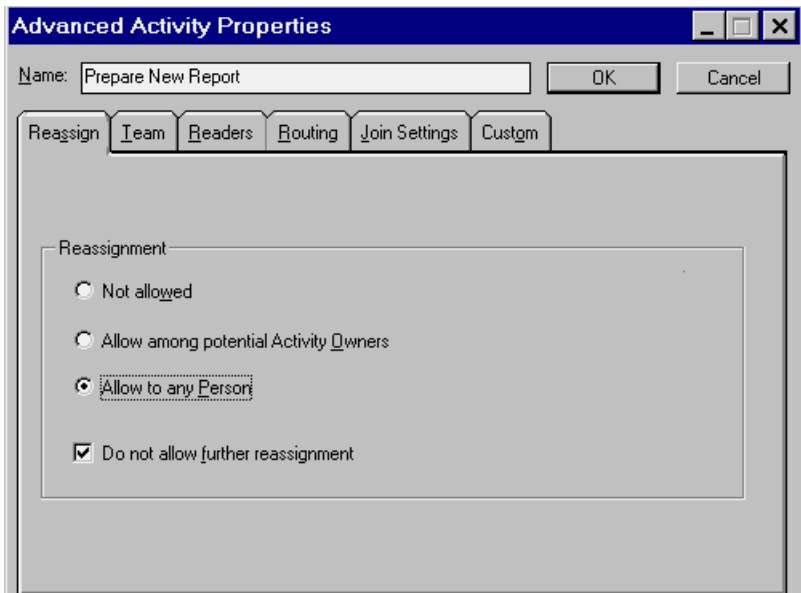
The screenshot shows a mail inquiry form with the following fields and options:

- from:** Karl Gutsze, 06.03.2000 19:16
- to:** A dropdown menu for selecting recipients, with two buttons: "Organization" and "Job context".
- Subject:** A text input field.
- Body:** A text input field.
- Options:**
 - Mail copy to yourself
 - Include doc link

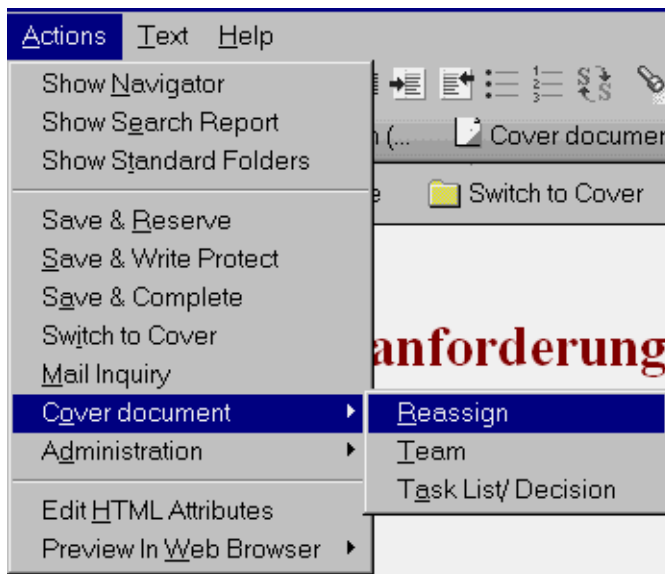
There is one more way of changing the team members. You can define an action on your document that allows you to set a new team. It requires changing the value of the internal workflow field named TeamAuthorsOS and TeamBackupOS in all binder documents to the user names of the people you want to have on the team. At the end of your code you can automatically create a mail inquiry and send it to the new team members.

Introduce some new activities (reassign)

In all processes that don't require very strict routing control you can allow the activity owners to reassign their work to someone else. To do this you'll have to allow for reassignment in the process definition (which is a default setting). You can find the reassignment settings in the advanced activity properties section in the Domino Workflow Architect (select an activity, right-click on it, and choose Advanced Properties).



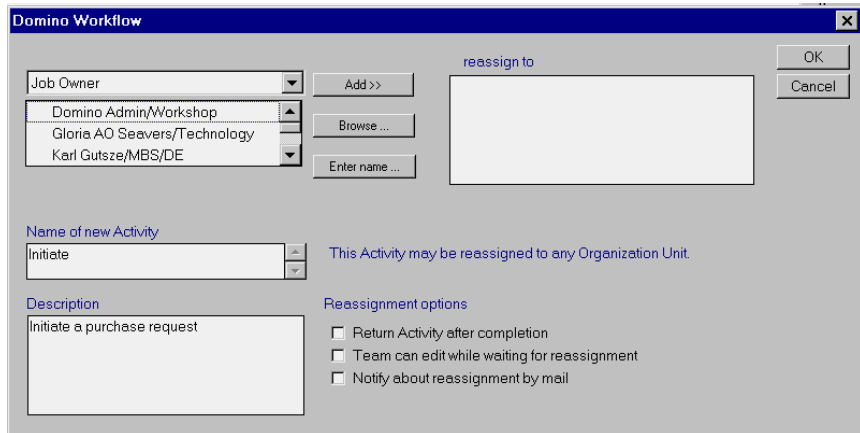
While working on your job you will have access to the Reassign feature only when you have one of the binder documents or the cover document open in Edit mode. Then you can choose Actions - Cover document - Reassign from the menu, as shown in the following figure.



From the cover document it is also possible to use the more directly accessible Reassign action button. The cover document has to be in Edit mode to make this action accessible.



When you choose reassignment you'll be presented with the screen that allows you to select the persons to whom the job will be reassigned, as well as other settings. As the reassignment allows you to create a dynamic quasi-activity, you can specify its name and give it a description. You can also specify that you want to work further on the same activity before it is completed, you can prevent further reassignment, and you can send mail notification about this assignment.



You remain an activity owner as long as nobody else claims the job. Up to this moment you are also able to withdraw your reassignment and to continue working on the binder. If a reassignment is accepted, the job's further path depends on the settings you've made.

Change the route of your job (reroute)

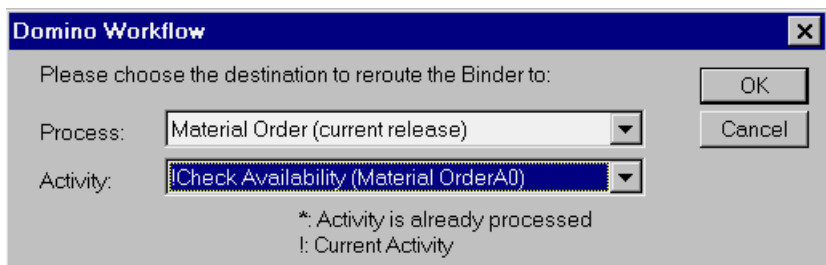
As a job owner you are able not only to reassign the current activity to some other person, but also to change the flow of the job entirely or to evaluate the potential activity owners one more time. Such a possibility can be very important for avoiding errors at run time and for testing your processes. To enable rerouting in your application you must enable the reroute option (the default setting is "disabled") in the application setup document (in the section "Global Settings") as shown in the following figure.



The Reroute action becomes available in the pull-down action menu in the view Desktop - 5. By Job Owner once you have enabled this option.

Note The only place where you can access the rerouting feature for a specific binder is in the view Desktop - 5. By Job Owner.

When you select the Reroute option from the action menu a dialog box will appear as shown in the following figure. It gives you the ability to move the binder into some other activity belonging to the same process, to change to a new process version, or to jump into some totally different process. It also allows you to evaluate the potential activity owners one more time. This capability is very important if some organizational changes occur and the members of the group that should work on a particular activity have changed. It may also be useful if someone gets sick and you are using surrogation rules in your company.

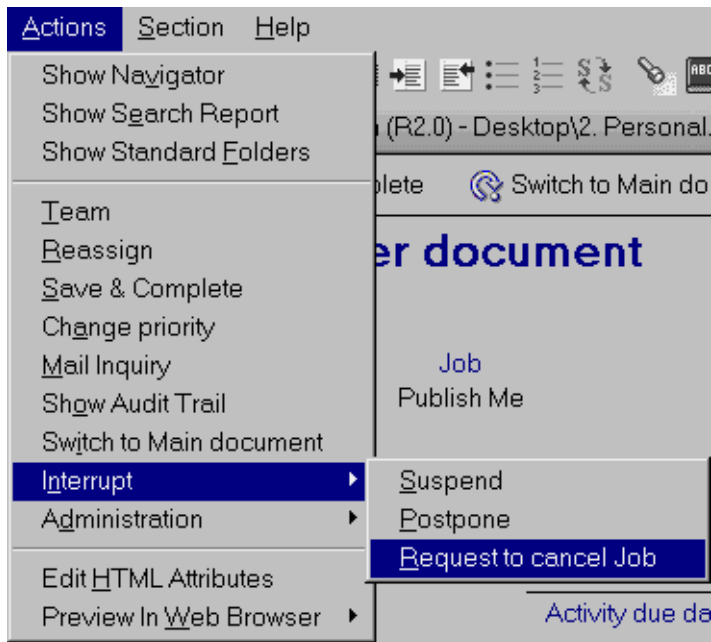


Therefore, the reroute feature allows you to update or migrate your processes in case of any changes that have to be done to the existing jobs.

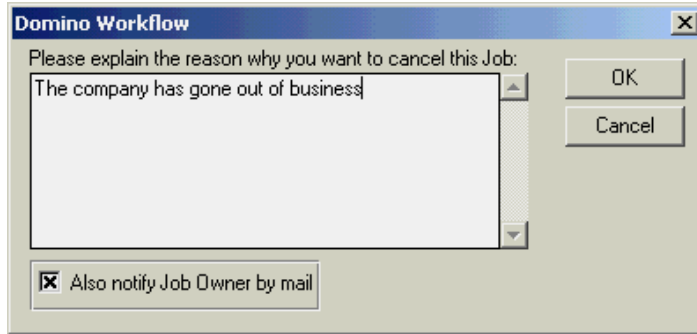
Cancel a job

There can be situations in which further routing doesn't make much sense and an activity owner decides to terminate the routing of a binder. Since an activity owner doesn't have enough authority to execute such an action (responsibility is extended only to one activity and not to the whole job), such an action requires help from the job owner. An activity owner has to open the cover document in Edit mode and then the appropriate action becomes available, as shown in the following figure.

Note In the same place you can suspend the further routing of the job or postpone it until a given date.



The following dialog box is shown after Request to cancel Job is selected.



Here the activity owner can explain why the job should be cancelled. This explanation is saved with the job and unless the activity owner deselects the option it also is sent to the job owner. The mail being sent to the job owner is shown in the following figure:



Afterwards a job owner has to decide if a job will be canceled or not. This is done by opening the cover document and selecting one of the available actions. A job owner can also cancel a job without receiving a request to do so, as it is the job owner's responsibility to take care of the overall execution of the jobs.

Keep an eye on the jobs

In many cases you want to know what happens to a job someone else is working on. It might be information that a customer requires or just a piece of statistical information. It might also be a way to check if everything goes smoothly in your processes or to track and trace a certain job.

Domino Workflow provides several features to provide you with this information. The most common mechanism is specific workflow views. This section describes some of the views you will find useful. The product also includes the Domino Workflow Viewer, a visual tool that gives you quick access to all the relevant information about a particular job.

The most important views on workflow

The most common mechanism used in Domino to display documents sorted on various criteria is views. Domino Workflow also uses this mechanism to provide you with some standard ways to look at your jobs. There are some views that you can use for the regular work, as well as views that allow you to keep an eye on the jobs. There are also some administrative views that make the maintenance and troubleshooting of your application easier. In this section we introduce some of the most important ways to look at your jobs. We won't try to explain all of them, as some are self-explaining and some others are rarely used. For complete details of all views see the product documentation.

Important You are likely to develop your own views and ways to display workflow jobs. Nevertheless, be careful with eliminating the provided views, as some functionality may be accessible only from them. We strongly recommend hiding standard views from the end-users instead of deleting them (at least until you are completely familiar with Domino Workflow and have used it operationally for some time).

What happens to a job?

In many scenarios you'll need to have access not only to your own work, but also to the work done by some other members of a specific group. To give you this ability you can define readers as an advanced process property in Domino Workflow Architect. This will allow you to view all jobs based on this process in the For Your Interest view that you can access through the navigator by selecting "Desktop."

Note The default setting for readers in the Architect is All, and such jobs are not displayed in the For Your Interest view. Only jobs with an explicit specification of readers will appear in the For Your Interest view.

Browsing through all jobs

One more easy way to find some jobs, or some kind of overview of them, is through the views that show all the jobs. If you know the name of the job you're looking for or if you use some naming convention, you can have direct access to a particular job through the view called By Job Name. If you want to have a look at the overdue jobs or the jobs that are close to being overdue, you can use the view By Due Date.

Note The view By Due Date makes sense only if the Timing setting was used in the basic process properties in the Architect. The due date shown in the view is a date on which a process is due, and not a due date for its activities. Timing settings for the activities have nothing to do with the timing setting for the process.

The next view after By Due Date sorts the jobs by status. This view is important for two reasons. The first reason is to show you how many new jobs exist in your application that have yet to be assigned to some person. This information lets you plan the work better. The second important reason concerns automated activities. You can determine from this view if the automated activities run without problems and if the jobs are routed further on.

Eliminating potential problems

The view sorting jobs by participant allows you to check the workload being put on all the workflow participants and to intervene in case of organizational changes. For instance, if someone had an accident and can't come to work for the next few weeks, you can check what jobs he already claimed (those jobs with a status of In Progress) and, as a job owner, redirect them to someone else. Such a change can be made by a job owner using the reroute feature, as described previously.

Finding old work

There are also views showing completed and archived jobs. These jobs can't be found in any other view for all jobs or desktop since they are no longer active and nobody needs to work on them. They might be of relevance to the customers, or needed for some statistical reason, and that is why you see them at all.

What happens in the application?

An application database contains not only views that allow access to the documents on which you work, it also contains views that help you to get your application up and running, and to check if everything is behaving according to expectations. Those are views reserved for the administration of your application. Therefore, you can find them in the pull-down menu under View - Administration. They not only allow you to have a look at various things, but also to take some actions that may be required.

Alternative initiation

Since the regular views show only active jobs, we need a place to look at the potential jobs that will be initiated through one of the alternative mechanisms. Such documents can be found in the view Administration - To be initiated. This is the place to look for all new documents that will initiate a job by the next execution of the appropriate agent.

It is also the place for documents that actually should initiate a job, but didn't do it for some reason. If you want to check jobs belong to this category, you can trigger the action called Mail-based initiation from this view. You need the appropriate rights on the server to run an agent triggered by this action. After you try to initiate new jobs, there should be no document in this view. If any document is left, you should check why it's still there. Among the items to check are that the process name can be found in the right place, and that all required fields are set. You can also check for errors in the view called Administration - Error log. A copy of each error notice can be found there.

Speeding up your tests

One of the administrative views looks very similar to the view for all jobs. It is a view for all jobs sorted by status. There is one big difference between those views: in the administrative view you can trigger an action calling the background agent. This agent takes care of all the routing in your application. Therefore, it may be useful for test purposes to run it after completing some activities. In this way you can force some transitions between activities without changing the client routing. (Like server routing, this is an advanced process property described in the product documentation.) You can also verify if a job is being routed further on or if it stays at one point of a process.

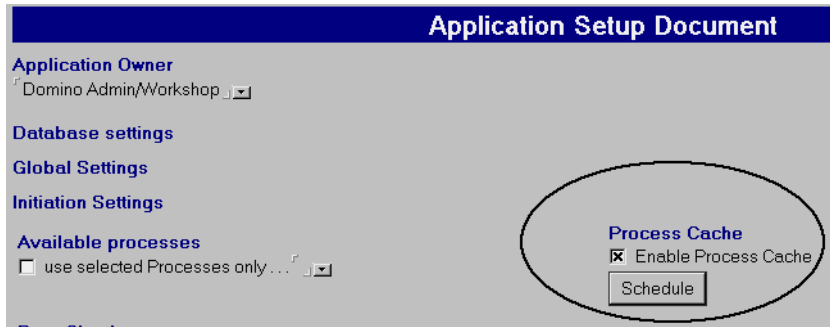
Note Always make sure that you have enough rights on the server to run restricted agents, and that you are assigned a role of process server in the application database before triggering a background agent. Otherwise, the correct execution of the agent will not be possible. You also have to make sure that the agent that you are running is not running scheduled on the server at the same time, as this could potentially cause many replication conflicts.

View all documents contained in the database

Another Administration view allows you to see all the documents in the application database sorted by their form. This is the only view in which all documents can be seen, and therefore it's a very important one for troubleshooting. If you have some problems with your application and you can't find any way of solving them, you can always go to this view, try to find a document that causes trouble, and analyze it thoughtfully. You can also consider deleting it and creating a document with similar content, but if you do so, make sure that this document is not a cover document or a main document in one of the jobs — otherwise you'll destroy the binder to which your document belongs.

Watch out for the cache

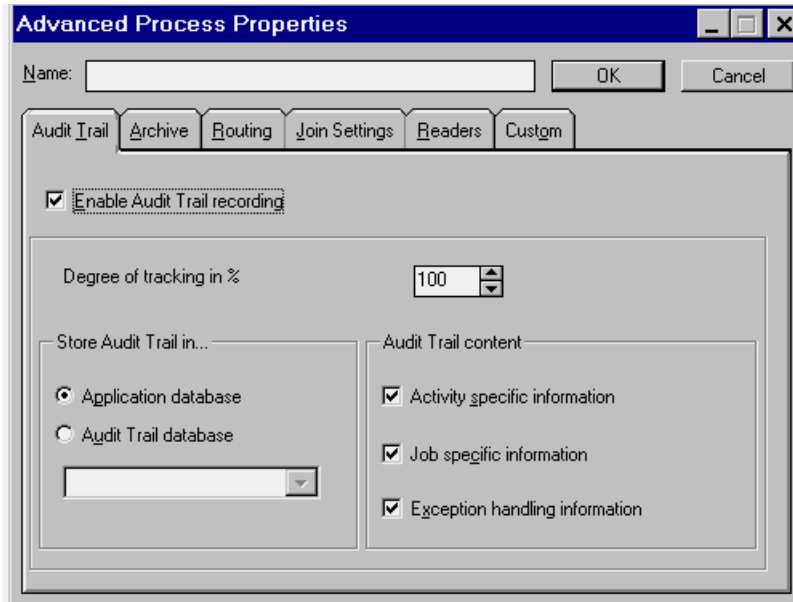
You may decide to use a process cache to allow quicker initiation of the processes. You can set this property in the application setup document in the initiation section. The default setting is without cache.



Enabling the cache allows you to eliminate the need to make a database lookup into the process definition database for each initiation of the process. All information about processes that can be initiated and people who can initiate them is stored in the application database and updated according to the schedule set in the application setup. A price for using the cache is, that the latest process changes may not be reflected in the cache. If a user was just added to a role or granted access to a process, this information may not be available in the cache right away. However, once a new job is started it always refers to the most current process version. In the regular environment such an inconsistency will rarely be a problem, as it lasts only a short time. However, it might be a problem in the testing and introductory phase of the process, where a lot of changes can happen in a short time. Therefore, you can use two views for monitoring your cache and updating it when required through a corresponding action.

Check all workflow actions (audit trail)

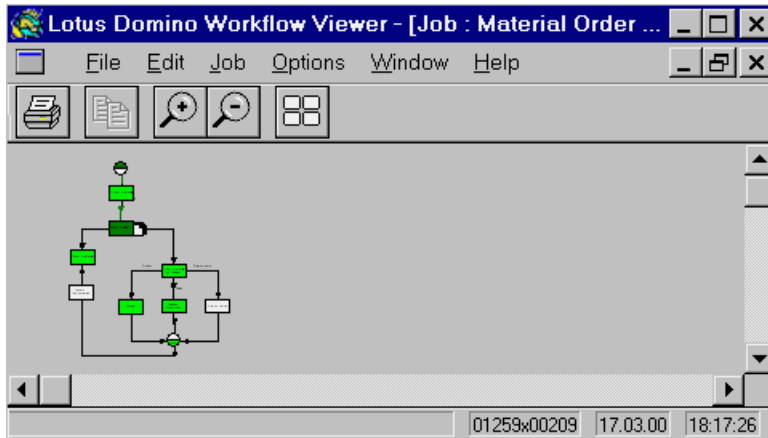
There is also one administrative view that doesn't show you the job documents at all, but only what happened to the job. This view is called Audit Trail, like all the documents that are displayed there. To have documents created for this purpose, an audit trail has to be enabled in the advanced process properties in Architect.



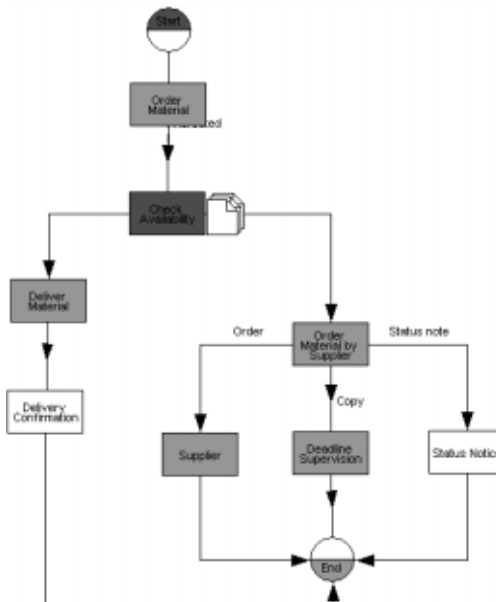
In the same place you can set the percentage of jobs that will be trailed, choose which kind of information should be registered, and choose a database in which all this information will be stored. More information about the audit trail follows later in this chapter.

Get all details of one specific job (process Viewer)

If you're interested in one particular job, its history, context, and possible outcome, the Domino Workflow Viewer should probably be your tool of choice. The Viewer is a visual tool that looks like the Architect, but allows you to see runtime information such as who initiated the job, or how long it took to complete a given activity. The Viewer can only read properties and does not allow you to change them. You can view a graphic representation of the job you're working on, see detailed information about the process the job is based on, determine who has already worked on the binder documents, and see who can claim the binder when you're done with the current activity. You can even preview the process design before you start a job.



As you can see on the following figure, it shows information that Architect doesn't show. It shows, for instance, the current position of a binder and what steps it has gone through. It is also possible to see who has worked on the binder and what exceptions took place.

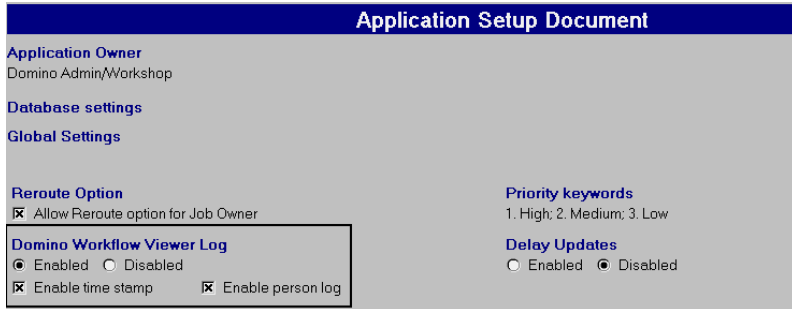


Note The Domino Workflow Viewer is the only part of Domino Workflow that has to be installed locally on the user's machine. However, the Viewer is an optional tool and is not required to use Domino Workflow. The installation process is quite easy and is supported through an assistant.

It happens automatically when a workflow user decides to view a job for the first time by using a corresponding action in the binder document or a cover document. You must be assigned the role of [Resource User] in the organization directory to be able to use a Viewer.

Note Viewer is stored in the organization directory as a resource. Since version 2.1 of Domino Workflow, Viewer is part of a standard installation; previous releases require a separate Viewer installation.

In order to be able to view a job, you also have to set appropriate properties in the application setup document.



Application Setup Document

Application Owner
Domino Admin/Workshop

Database settings

Global Settings

Reroute Option
 Allow Reroute option for Job Owner

Domino Workflow Viewer Log
 Enabled Disabled
 Enable time stamp Enable person log

Priority keywords
1. High; 2. Medium; 3. Low

Delay Updates
 Enabled Disabled

You have to enable the log, and specify if you want to record time of actions and the names of their executors. If you disable these options you'll gather less information, but your information will be clearly aimed at controlling a process without controlling the people. Such an approach might be appealing to companies that want to show that they trust their employees or those that may be required to do so for legal reasons.

More information about Viewer can be found in the product documentation and by following the links to the Business Partner Zone for Domino Workflow on the Lotus Web site at:

www.lotus.com

Recording your job context (audit trail)

As we mentioned previously, it is also possible to keep track of all major workflow actions using audit trail. It records actions that occur at the job or activity level, such as who claimed an activity, who received work reassignments, and the time and date work was completed. By analyzing the audit trail, you can discover bottlenecks in the process and other irregularities. Each action recorded is stored in a separate audit trail document. The audit trail can be stored, as well as archived, in separate databases. The audit trail must be enabled in the advanced process or activity properties before you can use it.

Note If you're using a separate database to store your audit trail, it must be a mail-in database in your Domino Directory and defined as a resource in the organization directory.

Using events to customize the audit trail

You can use the events included in the Domino Workflow Developer's Toolkit to extend the features of the audit trail.

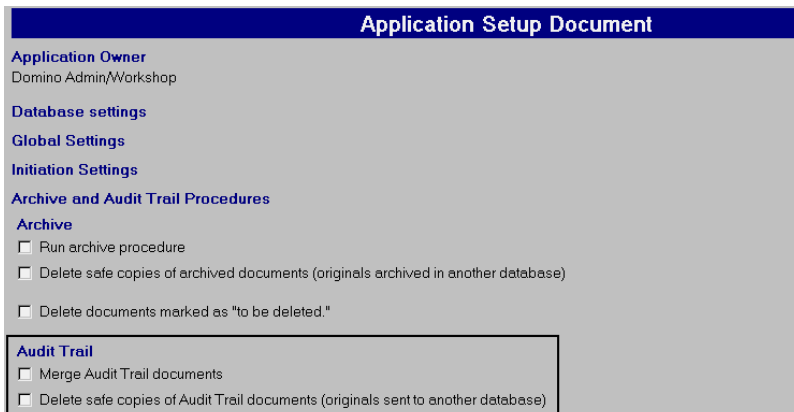
You can use audit events to append additional data to the audit document; for example, you can append the current team members, or perhaps the current state of other processes. Audit events can be used to extend and modify the audit document that is passed as a parameter in every audit event handler. They can also be used to perform actions you define, such as creating or modifying other documents.

Nearly half of the Domino Workflow events are audit events. Every audit event is a query event. You can set the Continue flag to False to prevent the system from creating an audit trail entry.

Important Audit events ("QueryAudit . . .") will be executed only if audit trail is enabled in the Domino Workflow Architect.

Managing the audit information

In addition to activating audit trail on the process and activity level, choosing the storage application on the process level, and using some events to customize the way audit trail works in a specific application database, you can set some properties for all processes in one application. These additional properties can be set in the application setup document.



The screenshot shows a window titled "Application Setup Document" with a dark blue header. The content area is light gray and contains several sections:

- Application Owner**: Domino Admin/Workshop
- Database settings**
- Global Settings**
- Initiation Settings**
- Archive and Audit Trail Procedures**
 - Archive**
 - Run archive procedure
 - Delete safe copies of archived documents (originals archived in another database)
 - Delete documents marked as "to be deleted."
 - Audit Trail**
 - Merge Audit Trail documents
 - Delete safe copies of Audit Trail documents (originals sent to another database)

Since audit trail creates one document for each action, there are a lot of documents created for one job. Therefore, in most cases you will want to merge all this information in one document. Domino Workflow can do it

automatically for you if you choose the appropriate setting in the application setup document. If you decide to store the audit trail in a separate database, an agent deletes the original documents from the application database after mailing a copy of them into the central audit trail database. If you activate any of those settings, you have to decide on what schedule the appropriate agent would run. To do this, use the Schedule button that appears on the right side after you enable any option.

A completed job (archive)

After the last activity in a job is completed, the whole job can be found in the view called Completed Jobs. It may stay there as long as the database exists, but it's also possible to archive it. With archiving you can prevent the application from growing too fast, and also remove old jobs from the operative server. You must determine what kind of archive is appropriate for your company; in this section we introduce the archiving mechanism offered by Domino Workflow.

Tip Always consider the use of archiving in your application database to keep your runtime environment as compact as possible. Performance problems can occur if your database is too large. Therefore, a general rule in developing Domino Workflow applications is to keep your application as compact as possible.

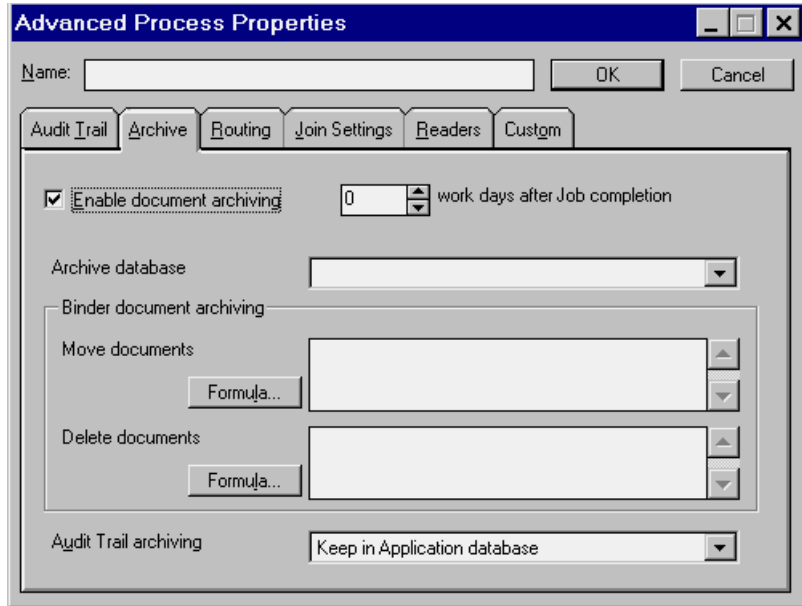
Setting up an archive

Prior to using archive in your application, you have to set some properties to enable it on the process level (in the Domino Workflow Architect) and on the application level (in the application setup document). In Domino Workflow Architect you can find the archiving settings on the advanced process properties tab. Use the following steps to set up your archive.

1. Enable archiving — by default archiving is turned off.
2. Decide *when* the documents should be archived. For example, you may want all documents to be accessible in the application database for a few days after completing a job in order to allow easy access to them.
3. Define the database into which the documents should be moved. It has to be a mail-in database defined in Domino Directory and it has to be declared as a resource in the organization directory. The archiving mechanism uses e-mail for transferring the documents into an archive.
4. Specify the documents that should be archived. You can use any formula that would work as a view selection formula for making such a choice. This setting allows you to archive only selected documents containing some important information. The next setting allows you to

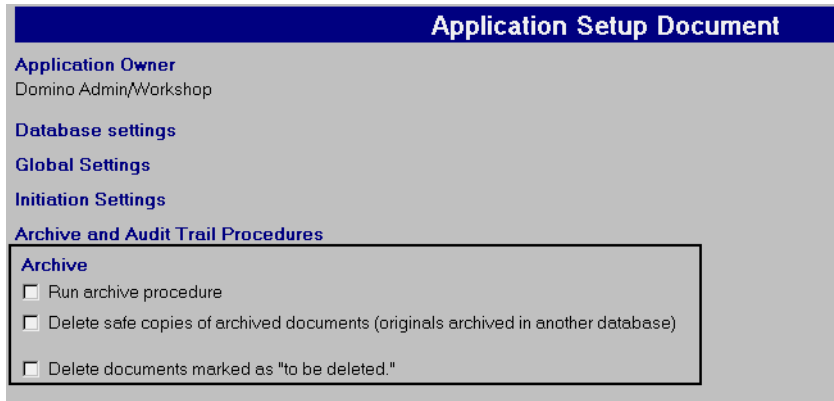
mark other, unimportant documents for deletion (documents that were archived don't have to be explicitly deleted).

5. The last setting on the Archive tab lets you specify what happens to the audit trail information of a completed job. It may be kept in the application database, moved into the archive database, or deleted.



After you've enabled archiving for any process, you also have to change some settings in the application database in order to make archiving work. To make these changes you have to open the application setup document and look into the archive and audit trail section. You can see the possible settings on the following figure.

- Enable the agent running the archive procedure (and schedule it using a button that appears on the right side as soon as you enable a setting).
- Choose whether to delete the archived documents (in the process settings you choose which documents should be moved and automatically marked as such). If you don't enable this setting, two copies of the archived document will exist: one in the application database and one in the archive database.
- The last setting allows you to schedule the deletion of documents marked for deletion according to your specifications in the process settings.



Various ways of archiving

The easiest way to archive your jobs is to use a standard archive database that comes with Domino Workflow. You just have to put it on a server, declare it a mail-in database in Domino Directory, create a corresponding resource in the organization directory, and set up archiving properties as described earlier in this chapter. This will provide you with a working environment that takes care of your basic archiving needs. However, you may want more from your system. You might consider some other scenarios for archiving completed jobs. Just as your scenario can be implemented with Domino, it can also be implemented with Domino Workflow.

Electronic document management systems (EDMS)

One common archiving scenario incorporates the use of an electronic document management system as a logical way to store your documents. You can connect your workflow with any EDMS by placing an automated activity on the end of your process, writing an agent that connects to a EDMS and checks all binder documents in, and by triggering this agent from the automated activity. In order to execute such an agent the information about your EDMS will have to be used. You can put it in the custom part of the application setup document, as the same information will probably be used for all processes. If you use Domino.Doc as your EDMS, you can set it up in the Domino.Doc Integration Setup part of your application setup document. You can also use an appropriate agent that is included in Domino Workflow (DD Archive), as Domino Workflow is already prepared for integration with this EDMS. There are other techniques to integrate Domino Workflow and Domino.Doc; they are described in the chapter about integration.

Archiving just the results

In many cases you won't be interested in archiving all the job information. It may be enough to extract some results produced in the job and some reference information. You can do this with a custom agent that will be triggered in the automated activity placed on the end of your routing path, or you can use some events provided in the Domino Workflow Developer's Toolkit. If you use Audit Trail you can place your code in the event `QueryAuditJobCompleted`, which offers you a hook to the binder cover, through which you can find the rest of the binder documents. In this way you can extract all the required information from the binder and, if desired, delete the documents. You can also use the events `QueryArchiveMove`, `QueryArchiveDelete`, and `QueryArchiveMarkPermanent`. These events are accessible if you've enabled the archiving functionality of Domino Workflow; they let you access separate binder documents. By writing some script into them you can also extract only the desired information, ignoring the rest of the job or binder content.

Archiving with stubs

One more archiving scenario may be considered very interesting. It is based on the assumption that you don't want to keep your jobs in your application any longer than required (for the obvious performance reasons) but want to be able to find all their information very quickly if required. To do this you will need some information in the application database that allows you to access a particular archived job. There are many ways to accomplish this, for instance by inserting your script into workflow events as described in the previous chapter, or by disabling default archiving and writing your own agent for this purpose. You can run this agent on all completed jobs that are found using the following formula:

```
SELECT InstanceIDOS != "" &(@UpperCase(FolderStatusOS) = "JOB COMPLETED")
```

Your agent should run on each binder, and on each binder document. It should perform the following actions:

1. Get the binder cover.
2. Create a new job reference document.
3. Copy some job information from the cover to a new job reference document.
4. Copy the binder cover to the archive database.
5. Create a link to the binder cover in the archive database.
6. Place the newly created link into the new job reference document.

7. For all binder documents:
 - a. Get document.
 - b. Create a copy of this document in the archive database.
 - c. Extract some document information.
 - d. Write this information into the job reference document.
 - e. Create a link to the archive document.
 - f. Place your link on the job reference document.
 - g. Delete the binder document.
8. Find the Audit Trail document for this job (if it exists).
9. Create a copy of this document in the archive database.
10. Create a link to the archive document.
11. Place your link in the job reference document.
12. Delete the Audit Trail document.
13. Save your job reference document.
14. Delete the cover document.

It is your choice what information will be saved in the reference document. It is also possible to create a link only to the main document or some other kind of document, but those decisions have to be made depending on your process scenarios. This discussion shows you only an overview of how you can easily customize your environment according to your needs.

Optimizing your workflow application

A Domino Workflow application is just a customized Domino database, so the optimizing information that applies to any other database also applies to your application. A good source of information about designing for performance is the IBM Redbook *Performance Considerations for Domino Applications*, SG24-5602-00. More information about this book is on the Web at <http://www.ibm.redbooks.com>.

All the optimizing issues that are important for Domino are also applicable for Domino Workflow. Nevertheless, we want to remind you of some issues that we view as critical for the safe deployment and successful performance of your workflow.

Design elements

To customize your application you might need design elements other than those included in Domino Workflow. Customization is easily done and causes no harm, as long as you don't change any elements that have OS or DWF in their names. You can change, for example, all the views that will be presented to the workflow participants and other elements of the user interface. However, you shouldn't change the hidden views that are used for accessing workflow information. You might add some graphics to the included subforms, but you should still preserve all the contained fields.

Hiding a cover document

Probably the most common customization of Domino Workflow is connected with hiding a cover document. In the cover document you can find a lot of information about a process and current activity, but not the content of your binder. Therefore, the real work happens mostly in the main document. A cover document can be considered the overhead that is required to put the binder into the right context so that it can be occasionally accessed, but without the need for it to be exposed all the time. You can create a new view that displays all the workflow documents, but doesn't contain cover documents. To do this you can just add one more restriction to the view selection formula, eliminating those documents. For the view showing all documents your selection formula will look like this:

```
SELECT InstanceIDOS != "" & @UpperCase(FolderStatusOS) != "JOB  
COMPLETED" & @UpperCase(coverdocOS) != "YES"
```

You'll still have access to the binder cover from the document using the appropriate action. For further customizing of the view, you might want to preserve the first three columns that categorize your jobs and you might leave the fourth column showing the status of the job. You can customize the rest of the view according to your needs.

Customizing actions and dialog boxes

Generally, it's no problem to customize workflow actions or dialog boxes, as long as you don't change the workflow references contained in them. It means that you can change the look and feel of the user interface as far as you want, as long as it doesn't change the logic of your workflow context. You can remove some actions from the workflow subforms if you don't want to make them accessible for the participants. You can also change names of the actions, but do not change the reference to the agents that they trigger. It is also possible to place your company's logo inside of the dialog boxes or to use the names that are more suitable for your company. You just have to remember not to change any workflow fields included in them.

Performance issues

In many companies the performance of Domino Workflow can be considered one of the most important issues. As Domino tends to have pretty long response time in some situations, you have to be even more aware of this problem in the case of an application based on Domino. It is of utmost importance to design your workflow application according to some basic rules to avoid slowing it down.

Minimizing the size of your application

One of the basic rules in designing a Domino application is keeping the size of it as small as possible. This general rule applies to a workflow application even more than to any regular one because of the overhead caused by the workflow information. In the next few paragraphs we briefly discuss some important strategies for minimizing application size.

Archive

We already discussed the methods for archiving in this chapter. Here we only want to stress one more time the importance of this technique. To keep your application in top condition we recommend strongly that you move completed jobs into some other application as soon as possible, and perform all aggregations and evaluations on the archive database. It is of no importance whether you use the provided mechanisms for archiving or implement your own archiving solution, as long as your application contains mainly current, running jobs and not the obsolete ones kept only for statistical or legal purposes.

Distribute your application

Another way to keep your application small is by dividing it into more than one database. There are two different possibilities for achieving this.

The first and obvious way is to create separate applications for each process (or group of processes). This approach works very well with high-volume processes where workflow participants work mainly on one process. Then each participant has an application database that acts as a main field of work, and in case of involvement in some other processes a participant should get a mail notification of the required involvement. This approach doesn't work if the participants have to work with multiple databases in which some work is sometimes assigned to them, as the process of looking for work can require more resources than the work itself.

In a situation in which the workflow participants can be clearly assigned to one server and the groups that are used in the process model are always using an application replica located on one server, you can try to reduce the size of the application by means of selective replication. However, the issue is complex which is why a separate chapter of this book (Chapter 9) is devoted to the distributed environment and replication issues.

Delete outdated data

You will probably create a lot of design elements and save copies during the implementation of a workflow application. When a development cycle is finished, you can delete all design elements that are no longer relevant. These elements will not completely disappear; their corresponding stubs will remain in the database. The stubs are needed to refresh replicas of the database when the deletions are made in them. If these stubs are no longer relevant, you should remove them from the database. Stubs can't be removed by compacting a database. However, they can be deleted by changing the replication settings under "Space Savers." Change the default setting from 90 days to 0 days. Deletion stubs are also created when deleting documents from a database. For each document, a deletion stub remains in the database. With the procedure described here, all these stubs will be deleted, too. To recapture the free space that removing these stubs has created you have to compact the database.

Setting up your application

It is not only the size of your application database that has impact on the performance. There are also some settings in the application database that you have to consider. Most features that automate some kind of product behavior also have some impact on the workflow performance because some conditions have to be checked, and if required, some action has to be taken. In most cases you probably will not use all of those features, and therefore you should disable them in order to reduce the overhead caused by the workflow. All the settings that we discuss in the next few paragraphs can be found in the application setup document that can be accessed from the pull-down menu View - Administration - 1. Application Setup.

Option subprocesses

Remote subprocesses — the task that moves binders into remote subprocesses (that is, they run in a different application database) — can be deactivated if your application database does not run processes that move binders into remote subprocesses. The runtime of the Domino Workflow Backgrounder Agent will be reduced if this option has been deactivated because less program code has to be executed. Disabling this option will still allow you to run subprocesses within the same application database that is running the host process.

Send mail notifications

This option can be deactivated if no e-mail notifications need to be sent to the potential activity owners during a job. Again, the runtime of the Domino Workflow Backgrounder Agent will be reduced if this option has been deactivated.

Dupe check

The Dupe check routine is used for interactive initiation to check whether a user-defined job name already exists within the application database. The time necessary for activating a new job will be reduced if this option has been deactivated. No check will be made whether a job name exists. If you can be sure that each job name will be unique, or you don't care about it, you can deactivate this option.

Process cache

The time necessary to start a new job will be reduced if this option has been activated. Make sure that you do not merely check the box; you also have to update the cache view in the application database under View – Administration – Cache, or make sure that the OS Administration agent runs regularly. If you want to use the application database on the Web, this option must be activated; otherwise, you won't be able to start a new job.

Routing options of the backgrounder agent

You can set the backgrounder agent to call the program code for routing and automated activity processing several times during each execution of the agent. Using this feature, you can reduce the amount of time that is necessary for one binder to route through several automated activities within a complex job. Keep in mind, however, that with an increasing number of routings the time for one run of a single backgrounder execution job will increase accordingly.

Note This effect has to be considered in the server document for the maximum execution time settings for the agent. If the time that is set there will exceed the new average runtime of the backgrounder agent, the corresponding value has to be increased accordingly.

Setting up the organization directory

Though we are focusing primarily on the concerns related to an application database, we should also consider a few settings in the organization database that have a very big impact on the performance of the whole system. These settings can be found in the setup document of the organization directory.

Work with surrogates

Domino Workflow offers you the possibility of using automatic surrogating mechanisms. If you have defined surrogate rules for an organizational unit, they will be used in jobs when no members of a specified unit are available. You can use surrogates not only for people, but also for departments, workgroups, and roles. If you don't use this feature, you should disable it. When substitute management has been disabled, the routing task takes less time because the engine does not need to check for Out Of Office Profiles and evaluate replacements.

Activate cache

When you activate this option, indirect members of groups and any substitutes will be pre-evaluated. In most cases, a significant performance enhancement during routing can be realized by this. The deeper the structure of the involved organization units, the greater the enhancement.

Routing schedules

Routing options and their related response times depend on many different factors. These include, for example, the complexity of the organization directory, the amount of data stored in the application database, and the size of the binders and documents belonging to them. For complex data, server-based routing is recommended. Routing of binders will be executed over the server. When an activity owner completes an activity, the binder will get the status "Ready to route". Only specific program modules run, so the document will close faster and the user can get back to work sooner.

To reduce the time intervals necessary for evaluation on the server, you can reduce the cycles of scheduled agents. These time parameters are set in the configuration document of the server. The corresponding numerical data should be set individually.

AMgr_DocUpdateAgentMinInterval = 0

This parameter indicates the minimum number of minutes that have to pass between executions of the agent that is triggered for the update.

AMgr_DocUpdateEventDelay = 0.

This parameter indicates the time delay in minutes, before the agent manager is triggered after an event that updates the document.

Note Setting these values at 0 still means the polling time is going to be two minutes.

The corresponding scheduled agents won't be activated with the option "scheduled," but they will be activated with the option "when documents are created or changed." This new setting will result in a more homogeneous as well as a reduced server load since the agent only has to execute the modified and new data. This results in a noticeable performance enhancement for the actual users.

Important Changing the Agent Manager settings has an impact not only on your workflow application but also on your entire server performance. You should always consider all possible impacts of such a change.

Troubleshooting

Domino Workflow is a complex software tool. Therefore, there are many possible error sources in case of any malfunction. It is impossible for us to anticipate and list all the possible errors and ways to avoid or fix them. What we can do is try to identify some very common errors and give some general advice on where to look for help. However, the best source of knowledge about the common errors, system limitations, and ways to handle them is the product documentation and the ongoing Domino Workflow discussion on the Web at:

<http://www.lotus.com/workflow>

in which the most common problems are discussed. In this forum you can also post questions, and possibly get direct answers from the product developers.

Some common errors

In the next few paragraphs we'll discuss some errors that can happen to everybody. Most of them are easy to repair, and therefore worth mentioning for quick reference. This should also show you some potential error sources and some general ways to handle them.

Errors when activating a process

The first time you can encounter errors with your process is when activating it. There are mainly two reasons for problems at this point, and both of them are easy to identify. They are also relatively easy to correct.

The first possible cause of an error when activating a process is due to insufficient access rights on the process definition database. As you try to activate a process, Domino Workflow involves you in a dialog where you have to choose in which database a new process should be activated. If you choose a database that is not a process definition database, or if you don't have appropriate access to write information into it, Domino Workflow generates an error notice. You have to correct the error and try to activate your process once more.

The second possible cause of an error while activating your process is a consequence of logical errors in the process. Domino Workflow performs a syntax check by activating a process. If any errors are found, they are all listed in a syntax check box, together with a description, and they're also highlighted in the process map to help you to locate them. You can activate your process only after you correct all the errors. You can also perform a syntax check without activating a process, using an appropriate command from the menu.

Problems while initiating a job

After activating your new process you might still encounter problems when creating a job based on its definition. Here we describe briefly a few of the most common job-initiation problems.

Not listed as initiator in the process definition

If you have activated a process and can't start a job based on it, the first thing to check is whether your process definition allows you to start it. You can initiate a job if you are one of the potential activity owners of the first activity or if everybody is allowed to initiate a job. You can check both those settings in the Architect. However, you must be sure your current version was activated. Therefore, it's usually easier to check your process in the process definition database. There you can check who can initiate a job and be sure that those settings are active.

False database configuration

If you've verified that you should be able to activate a job based on a specific process but it's still not possible, check whether your databases are correctly set up. You can do this by checking the paths in the application setup document. If you have multiple servers, you also have to check the server. If you have multiple replicas on one server, check that they contain the same information. You can also check the configuration, looking at the availability of some other processes.

Only listed processes can be initiated

Database configuration may cause some problems during the first process. If you already have at least one running process in your working environment, the introduction of the next processes shouldn't cause any trouble, as the configuration doesn't change. Problems can be caused, however, if you restrict the processes to only a few explicitly listed in the application setup document. In such a case you should change the application setup document (which you can access from the pull-down menu View - Administration - 1. Application Setup) in the Initiation Settings section, under the first item, which relates to Available processes.

Process cache activated

In some cases you can use the process cache in the application database. The cache can be used for performance reasons and *must* be used if you want to allow process initiation from the Web browser. In such a case there is some delay between activating a process and being able to deploy it. When testing, or if it is necessary to work with a process as soon as possible, you will need to update the cache manually. Otherwise, you have to wait until the scheduled cache update takes place. To update the cache, open the view Administration - 7. Cache - 1. By Initiator from the pull-down menu and trigger the appropriate action by clicking Update Process Cache. Afterwards the names of all people that can initiate a job will be listed, together with the processes that are available to them.

Note A good aid for troubleshooting problems related to initiation is to populate the Process Cache even though it may not be used by the application. Open the view Administration 7. Cache - 1. By Initiator or ... - 2. By Process. If you have the appropriate access rights (database role [Process Cache]), you will be able to use the view action "Update Cache." After using it, a document will be created for each process that can be started within this database. You can review who is allowed to start these processes or whether there are any other problems.

Routing problems

Most problems happen after a job is initiated: there are all kinds of errors possible at this time. We'll describe here only a few of the more common ones. It is not possible to come up with a complete list of all possible errors, as they can be caused by false process logic, problems with your application, or problems with Domino in general. Therefore, we show only some errors typical for Domino Workflow. They all cause a situation in which a binder can't be forwarded after completing an activity.

Next participant can't be evaluated

The logical consequence of problems with evaluation of the next participant is the lack of routing. If the workflow engine doesn't know who should work on the next activity, it can't pass the work any further. If such a problem occurs during a client routing, the participant of the completed activity is presented with a screen informing them about the situation and offering a choice of people who can be sent a message. If a server routing causes the error, a job owner gets a message. In both cases the binder stays with the last activity owner and is shown in a view with an additional comment about a routing error.

Such a situation can be caused by changes in your organizational structure such as deleting a group or removing all its members, by removing some members from your organization, or by creating some out of office profiles without defining surrogates. The other possible source of the error is routing using a formula or field value for evaluating workflow participants that doesn't evaluate to the people listed in the organization directory.

In any of these cases a job owner (in some cases also an activity owner) has to check what caused the trouble and then reroute the activity as described earlier in this chapter. A reroute should follow the activity following the completed one. This strategy should let you solve a problem relatively quickly and should let you continue workflow as soon as possible.

All conditional paths evaluate to false

There is also a possibility that the binder cannot be forwarded because of the lack of a routing path leading to the next activity. Such a situation can happen if you use conditional routing without using an else connector. If you change some of your fields or the context of your process changes

somewhat, what can easily happen is that all conditions will then evaluate to false and no further routing is possible. For Domino Workflow this situation is just like the last one described: the next activity owner can't be evaluated. An error like this can't be fixed by a job owner other than by using the reroute feature, it has to be resolved by a person responsible for designing the process. Therefore, you should always have an else connector coming out of any activities with only conditional routing relations. In this way you are always on the safe side with minor design changes because a default routing path exists.

No further routing by an automated activity

Automated activities always carry a potential risk. If a job gets stuck in them you don't always receive an error notification and no workflow participant can see the problem directly in the work environment. Therefore, it's important for a job owner to look into the view showing the status of all jobs from time to time and to check the automated activities periodically. If an automated activity should send a mail notification, there is no problem. If it sends a mail, the binder is routed further, and if it can't send mail, a job owner is notified. It's more complicated when running a LotusScript agent. An automated activity should be passed on further after the completion of the script. Because of this, if some errors in the script execution occur, the activity is not ready for further routing. It waits for the next cycle of the executing agent to complete the script. If the completion never happens, the activity will never be completed. Therefore, it's the responsibility of a job owner to take care of such activities and to notify process designers.

General tips

There are also some general problems that may occur during workflow execution. Some of these problems generate an error notification, others do not. In the next section we want to show you a few such problems.

Job owner not available any more

In some situations it's possible that a job owner is not available any more. This problem can be caused by the fact that the group of job owners is evaluated down to the individual people by the job initiation. The owners of the job stay the same to the end of a job. Therefore, if your processes last for some time and the environment changes, the probability of personnel changes grows larger. Unfortunately there is no explicit feature to change the job owner. Domino Workflow offers you no mechanisms for doing it. However, given appropriate access rights, you can write a small agent that changes the field JobOwnerOS in all binder documents. Please take care to use canonical usernames for all fields containing participant names.

Save conflicts

If a save conflict of any kind is generated in Domino Workflow, it can only be resolved manually. This is the same approach that Domino takes. You'll probably assign this job to a job owner, the person with the best overall knowledge of the process. It is necessary to transfer the whole content into one document and to delete the additional document. However, you can encourage your workflow participants to use locking mechanisms on the document level in order to avoid such situations. If your team members (save conflicts are typically created by teams rather than individuals) are working on the same server, locking should eliminate the problem.

Missing binder parts

There can be some situations where binder parts are missing. Such a situation may not happen without these documents being manually deleted. If the Access Control list is set properly users are not able to delete Cover documents. However, if they are Activity Owners they may delete binder documents. If this happens accidentally there is no undo mechanism. However, you may be able to replace the original document. You can do so by adding a new binder document or by pasting a similar document using the standard copy and paste mechanism. If for some reason the Cover document was deleted (e.g., by an automated activity agent that contained a bug), there is no way to recover this document. There are two alternative procedures. You can start the job from scratch and reroute it to the stage in which it became inconsistent. If the old binder documents are still consistent, you can cut and paste them into the newly created job. If the cover becomes missing in a parallel path of a workflow, you may be able to skip this path and continue at the next join. Both procedures cannot be applied universally and can only be used with caution. Whenever skipping workflow activities the data model may become inconsistent (for example, a decision is skipped and then missing later on in the process). There may also be interdependencies with external applications such as relational databases or Domino.Doc file cabinets used by the process.

As you can see, it's not easy to correct such an error, so it is best to avoid it. by setting the Access Control list properly and by educating your users.

Summary

In this chapter we wanted to share some thoughts about running a process and about mechanisms that can help you to take better care of your processes. During the real implementation of your processes you'll probably encounter additional difficulties, since your processes will be getting more and more complicated and your organization will depend more on workflow. At the same time you'll become more accustomed to Domino Workflow and you'll learn to avoid making some simple errors. You'll also learn to use efficiently some advanced techniques and features to control your workflows. However, you should always keep in mind that workflow is only one part of your company and its goal is to help people to get the job done. Your focus should be on business processes and people who work on them. If it sometimes means that you should apply less control and, in some cases, stop using workflow altogether or just embed workflow into some other context — this is what you should do. Workflow is a great tool to help you make your company more efficient and Domino Workflow is a great workflow tool — but it's no more than that. Your company has to create efficient business processes and execute them. That is the core concept behind the workflow idea.

Chapter 12

Domino Workflow and LotusScript

Not all coding in Domino Workflow can be done with @Formulas. If you want to run an agent in an automatic activity or need to react to Domino Workflow events, you need LotusScript.

In this chapter we will discuss:

- How to create agents for automated activities
- Classes for encapsulating Domino Workflow documents
- What to do with repeatedly failing automated activities
- How to create your own event handlers

Automated activities

Domino Workflow provides an opportunity to run a server-based task as part of a process using automated activities. There are three types of automated activities:

- Send mail
- Run a server-based program in an Operating System shell
- Run a LotusScript agent

The automated activity that runs an automation agent will be discussed in this section. See Chapter 4 for details about the send mail activity, and Chapter 7 to learn about server-based programs.

Automation agents

Automation agents can do a variety of tasks, for example:

- Collect data from other documents into the main document of the binder
- Create additional documents in the binder
- Wait until a certain condition becomes true

How to create an automation agent

To make it easier for you to develop automated activity agents, and because agents need to be built according to a specific structure, Domino Workflow provides an automation agent template.

The agent '**OS Automation template**' contains one function:

```
Function CustomSubroutine (binder As OSDocumentCollection,  
cover As notesdocument, main As notesdocument) As Integer
```

This function is all you need. It has three parameters, defined in the following table.

<i>Name</i>	<i>Type</i>	<i>Description</i>
binder	OSDocumentCollection	A collection of all the NotesDocuments that are in this binder (including the Cover and Main document). It has the same methods as the NotesDocumentCollection.
cover	NotesDocument	NotesDocument object containing the cover document.
main	NotesDocument	NotesDocument object containing the main document.

To create an agent for an automated activity you should do the following:

1. Make a copy of the '**OS Automation Template**' agent.
2. Open the copy and give it the correct name. The new agent name must not start with "OS" or "DWF."
3. Go to the CustomSubRoutine function and add your LotusScript code to it.

Note The result of the CustomSubRoutine must be a True or False value in the variable CustomSubRoutine.

4. Save and close the agent.

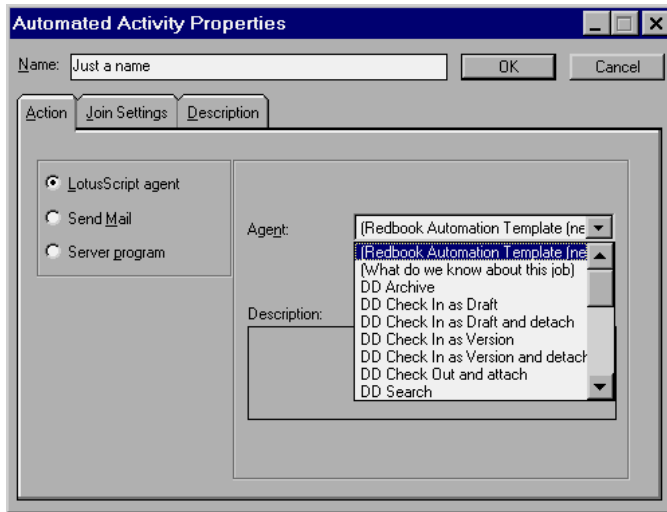
Now you are ready to use the agent in the Domino Workflow Architect.

Adding the agent to the activity

When using automated activities which run LotusScript agents you need two main things: an automation agent based on a template provided by Domino Workflow, and an automated activity in your process relating to this automation agent.

In the dialog box of the properties of the pertinent activity, use the following steps to add the agent:

1. Select LotusScript agent as type.
2. Select your agent from the list (see the following figure).



What happens when an agent is run

The Domino Workflow background agent will trigger your agent if there is a binder in an automated activity with your agent name. Your agent itself will (in the Initialize subroutine) determine which binders it actually will have to process. For each binder in turn the CustomSubRoutine of your agent is called. The return value of this routine determines if the binder has been processed successfully or not.

<i>Return value of CustomSubRoutine</i>	<i>Action taken</i>
False	The binder is not changed. Its status remains "Automation," so at the next run of the Domino Workflow background agent, it will run your agent again on this binder.
True	The status of the binder is set to "Ready to route." The Domino Workflow background agent will route it to the next activity.

Redbook Binder classes

If you want to refer to the Domino Workflow internal fields in your agents you will need to know which field contains what information. An object-oriented approach to interact with the Domino Workflow documents makes it possible to hide almost all Domino Workflow fields from your own code. This makes your code less dependent on the internal structure of the Domino Workflow documents.

Note These classes use the Domino Workflow Developer's Toolkit to process all the documents of the binder (Cover, Main, and additional).

Important The classes that we introduce in this book are not part of Domino Workflow and are only an example of how to encapsulate the data from the Domino Workflow fields. These classes do not necessarily reflect the object model underlying the future design of Domino Workflow. To distinguish our classes from the product code, we decided to use the prefix Redbook.

The library "Redbook Binder" contains two classes which can be instantiated by you.

- RedbookBinder
- RedbookBinderDocument

There are more classes in the library, but those will be used only by the RedbookBinder class itself. The description of these private classes is included in the description of the RedbookBinder class.

Binder

The RedbookBinder class encapsulates a Domino Workflow binder. With this class you have access to various properties of a binder. Properties which provide information about the section of the binder are put into groups like activity, job, process, or automation. The Domino Workflow Developer's Toolkit functions are used to find the documents of a binder. See Chapter 7 for more information about the toolkit.

Binder document

The RedbookBinderDocument class encapsulates the documents of a Domino Workflow binder. The class provides access to the common properties of a binder document like the identifier, when was the last Domino Workflow-related edit, and what is the status of the document. See Appendix C for details.

Extending the classes

The goal of these classes is to shield your code as much from the internal structure of Domino Workflow documents as possible. You can extend these classes if you need access to additional Domino Workflow fields. The subclasses RedbookActivity, RedbookJob, RedbookAutomation and RedbookProcess can be used to add, respectively, Activity-, Job-, Automation-, and Process-related information. You could also use the Redbook binder classes as base classes to encapsulate your own main document fields.

Redbook automation agent class

Each time you create a new automation agent you will have to start with a copy of the **'OS Automation Template'** agent. If you don't want to copy the whole template each time, you will have to put the code of the initialize subroutine in a shared place. One of the options is to create an automation agent class and use that as a base class for your own agents.

For this Redbook we have provided the RedbookAutomationAgent class, which uses the RedbookBinder class to do the search for the binder and its documents.

Redbook automation agent

The RedbookAutomationAgent class provides an easy way for you to create automated activity agents.

How to use

The RedbookAutomationAgent class can be used to create your own automation agents. To help you we have created an **'Redbook Automation Template'** agent. This agent contains a class with only one method:

```
Class MyAgent As RedbookAutomationAgent
    Function CustomSubroutine As Integer
    End Function
End Class
```

You just have to add your code to the CustomSubRoutine function, in the same way as you would have done it using the **'OS Automation template.'** You have already available:

<i>Name</i>	<i>Type</i>	<i>Description</i>
Binder	RedbookBinder	Object containing the current binder. It will give you access to most binder attributes and documents.
Agent	NotesAgent	Object containing the current agent.
Session	NotesSession	Object containing the current session.

You can define any additional functions or use additional libraries.

Note Make sure that you have CustomSubRoutine return True or False.

Caution The automation agents are run indirectly — that means they run within another agent (the Domino Workflow background agent in this case). For agents which are run indirectly, the case of the library name used in the 'Use' statement has to match with the name of the LotusScript library, otherwise the library won't load.

Examples of automation agents

Wait until next month

```
Function CustomSubroutine As Integer

    Dim First As Variant

    ' get the date from the main document
    First = Binder.Main.Note.FirstOfTheMonth(0)

    If first < Now Then

        'the date has arrived
        CustomSubRoutine = True

    Else

        'deactivate the job till this date
        Binder.Automation.Wait.Till = First
        CustomSubRoutine = False

    End If

End Function
```

Send the job owners a memo if a business rules check fails

```
Function CustomSubroutine As Integer

    Dim doc As NotesDocument

    Dim Body As NotesRichTextItem

    Dim MyDoc As New MyDocClass(Binder.Main.Note)

    Dim BusinessRules As Integer

    'the businessrulescheck returns true if the
    'document complied to the business rules
    BusinessRules = MyDoc.BusinessRulesCheck

    If Not BusinessRules Then

        'create the memo for the job owner

        Set doc = New
NotesDocument(Binder.ParentDatabase)

        doc.SendTo = Binder.Job.Owner

        Doc.Subject = "business rules check failed"

        Set Body = doc.CreateRichTextItem("Body")

    End If

End Function
```



```

        Body.AppendText("Business rules check failed
for '" & Binder.Job.Name & "' (main doc ")

        Call Body.AppendDocLink(Binder.Main.Note,
"Main document")

        Body.AppendText(")")

        call Doc.send(False)

    End If

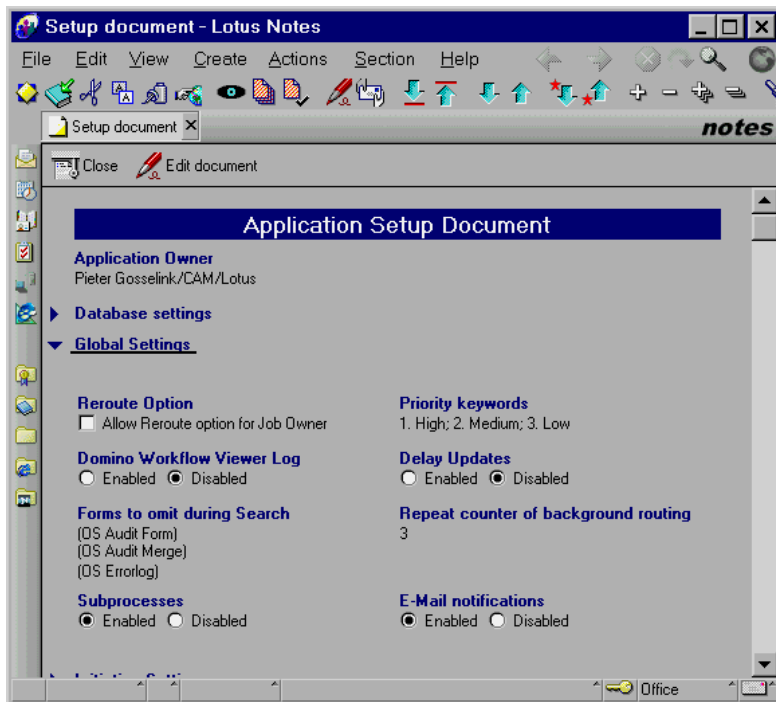
End Function

```

Unsuccessful automation agents

If the subroutine of an agent returns False, the agent will run again the next time automated activities are run by the Domino Workflow background agent. This can be more than once during one run of the Domino Workflow background agent because you can specify in the application setup how many times it should route activities. This means that unsuccessful agents will run as often as specified in the application setup document at each run of the Domino Workflow background agent.

See the option "Repeat counter of background routing" in the following figure.



This can become a problem if you have automated activities which fail repeatedly. An example is an automated activity which halts a job until a certain date. These jobs will be handled by the Domino Workflow background agent each time it runs.

This can mean that the Domino Workflow background agent has to run for a long time. Suppose it takes 1 second for each waiting activity (as mentioned previously) to determine that its date has not been reached yet. If you have 300 of these jobs active with the application setup as above, the Domino Workflow background agent will need $300 * 3 * 1$ seconds just for the waiting jobs. This is already 15 minutes.

Job deactivation

One of the solutions to these failing jobs is to run them at a lower priority. Domino Workflow, however, doesn't differentiate the agent execution according to the job priority. Therefore, you may want to hide some failing jobs from the Domino Workflow background agent. This can be done by setting the status of the binder to a value which is not recognized by the Domino Workflow background agent. The RedbookBinder class has the Deactivate method, which will prefix the status value with an asterisk (*). The binder can be restored with the RedbookBinder.Activate method.

The Run method of RedbookAutomationAgent class has two parameters:

- Increment
- Maximum

When an agent fails, a date is set on which it should be run again. For the first time an agent fails this would be 1 hour from that moment. Every time it fails again the interval is increased by a certain percentage, as specified with the Increment parameter. However, the interval will never be larger than the Maximum parameter. For example, if an agent is run with an increment equal to 50 and a maximum equal to 24 and the agent fails repeatedly, the intervals it waits will be:

1, 2, 2, 3, 5, 8, 11, 17 and 24 hours.

Note If you don't need to deactivate jobs, just specify 0 as the maximum.

Job activation

The disabled jobs have to be activated again when it is their time. This is done through a separate agent 'Redbook Activate Waiting Jobs,' which will activate jobs that have waited long enough. It will use the view '(Deactivated Automation Jobs)' which is sorted on activation time to determine which jobs should be activated.

Events

The Domino Workflow Developer's Toolkit provides a number of functions to access the Domino Workflow document hooks for numerous events that are generated by the Domino Workflow code (background agent, time management agent, and the client). You can write code which is run when the event occurs. For this purpose the library "OS Application Events" is included. All events are available in the form of subroutines that are called by the Domino Workflow code at the right moment.

Due to the fact that the library is already used by this Domino Workflow code, precaution should be taken when modifying the code of the "OS Application Events" library.

Changes to the event library

Some changes to the event library are permitted, while some others are not allowed.

Not allowed

The following modifications are not allowed:

- Modifying existing elements from the declaration section
- Removing any existing functions
- Changing the parameter lists of the existing functions

Allowed

The following modifications are allowed:

- Adding new functions
- Using additional libraries
- Adding elements to the declaration section

The Redbook Binder library can be used since it doesn't conflict with the existing libraries and it doesn't load the 'OS Backend 32Bit' library explicitly, otherwise a circular reference would have existed.

Refer to the Domino Workflow Developer's Toolkit documentation for a complete list of all possible events.

Note If you don't want to use events with all activities or processes, you can use custom attributes to select jobs or activities you want to work with.

Caution Don't declare Notes user-interface objects like NotesUIWorkspace in your code if they have to be used in the back end (like the 'OS Application Events' library). If you declare a NotesUIWorkspace in a scheduled agent or in one of the libraries it uses, the agent won't run. Some of the events have a NotesUIWorkspace parameter which is declared as a variant. You can use that in the front end.

Examples of event handling

Event: notification of an unclaimed, overdue activity

In this example we are going to make the job owner the activity owner of an overdue, unclaimed activity.

```
Sub QueryInboxOverDueMail(Continue As Integer, CoverDocument As NotesDocument)
```

```
    Dim Binder As New RedbookBinder(CoverDocument)
```

```
    Dim ErrorCode As Integer
```

```
    Dim ErrorMessage As String
```

```
    Dim success As Integer
```

```
    success = Binder.Activity.Claim(Binder.Job.Owner(0), ErrorCode)
```

```
    If Not success Then
```

```
        Print Str$(ErrorCode)
```

```
        Exit Sub
```

```
    End If
```

```
    success = Binder.Activity.Reassign(Binder.Job.Owner(0), False, True, True, Binder.Activity.Name & " (Escalated)", Binder.Activity.Description, ErrorMessage, ErrorCode)
```

```
    If Not success Then
```

```
        Print Str$(ErrorCode) + ":" + ErrorMessage
```

```
    End If
```

```
    Continue = False
```

```
End Sub
```

Event: notification after claiming an activity

In this example, we send a memo to inform the job owners that an activity has been claimed:

```
Sub PostClaim(BinderDocument As NotesDocument, Workspace As Variant)
```

```
    Dim Binder As New RedbookBinder(BinderDocument)
```

```

    Dim doc As NotesDocument
    Set doc = New NotesDocument(Binder.ParentDatabase)
    doc.sendto = Binder.Job.Owner
    doc.Subject = Binder.Job.Name + " claimed by " +
Binder.Activity.Owner
    Call doc.send(False)
End Sub

```

Event: In-box notification

Change the in-box notification memo so that it includes the database title in the subject, the activity description (if available), and the job name.

```

Sub QueryInboxMailNotification(Continue As Integer, ParentDB
As NotesDatabase, CoverDoc As NotesDocument, MainDoc As
NotesDocument, Subject As String, Body As Variant)

    Dim Binder As New RedbookBinder(coverdoc)
    MessageBox "inbox mail not."
    Call Body.AppendText("The job '" & Binder.Job.Name & "'
has arrived in your inbox.")
    Call Body.addNewline(1)
    Call Body.AppendText("You have to complete the activity:
")
    If Binder.Activity.Description = "" Then
        Call Body.AppendText(Binder.Activity.Name)
    Else
        Call Body.AppendText(Binder.Activity.Description)
    End If
    Call Body.addNewline(1)
    Subject = "A new activity has arrived in " &
Binder.ParentDatabase.Title
    Continue = False
End Sub

```

Summary

In this chapter we covered automated activities and event handlers. We showed how you could create agents to be run as automated activities. We have seen that it is possible to create classes which wrap around both the Domino Workflow binder documents and the Domino Workflow Developer's Toolkit functions. These classes can be used in both automation agents and event-handling routines. The class for creating automated activity LotusScript agents gives the ability to deactivate in case of a failure. In this way repeatedly failing jobs won't clog the system.

Appendix A

Parent-child processing

In Chapter 7 we discussed the concept of parent-child processing but did not give any details on how to make it work. Because this concept will be used in many processes, this appendix is included to provide some tips on how to successfully implement it, as well as some LotusScript code used in the two required agents.

Parent-child processing is not a term that you will find in the documentation of Domino Workflow. It is a concept allowing 1:n relationships among processes with a variable n (unlike standard Domino Workflow that requires the knowledge of maximum n in order to model a process). It also allows you to synchronize various processes and allows for communication among various jobs. It is an extension of standard concepts used within Domino Workflow. The standard concepts used for the implementation of parent-child processing are also described in this appendix.

Scenario

Imagine a workflow that is aimed at creation of a certain document, for instance a White Paper. The parent process will guide the user through the process, but at a certain time there should be a review of the document. This review can be done by one or more persons, depending on the content of the paper, its importance, and available resources. The number of reviewers usually varies from one to five, but in some cases can be as high as 20 or even more. Therefore, we have to construct a 1:n relation and we need parent-child processing in order to do this.

Concepts and features used

The parent-child concept is a combination of a number of concepts that you can find discussed in this book:

1. Automated activities and agents

To “spawn” the child processes you need an automated activity in the parent job that initiates all the required child jobs. In this automated activity an agent will run to create documents for each child job.

To “post back” all the information gathered in the child job, you also need an automated activity, this time in the child job. In this automated activity an agent will run to pass new information to the parent job and adjust the status of the parent job.

2. Form-based initiation

The documents created by the “spawn” agent will initiate the child processes (using the background agent for the form-based initiation). Your agent just needs to create the required number of documents including the five fields from the “Form based initiation template” and give those fields the right value.

3. Binder internals

Some understanding of the binder internals (items the engine stores and dynamically updates in the binder documents) is also required in order to understand the implementation of this scenario. You should also know how the engine keeps a binder together.

Parent-child processing uses all those features of Domino Workflow to extend the standard product with a new feature. It shows how you can satisfy your custom requirements by combining existing features with your own expertise into a single concept.

Steps to make it work

All the following steps are related to your Domino Workflow application database and Domino Workflow Architect. Two agents have to be created in the application database. Both new agents have to be based on the “(OS Automation Template)” that is a part of basic Domino Workflow (included in the product). Custom code that should be placed in these agents is provided in this section.

Follow these steps to make it work:

1. Create a “(Parent Spawn)” agent which will spawn the child processes.
2. Create a “(Child Post Back)” agent which posts back the information gathered in the child process.
3. Create the following control fields on the main form:
 - ExpectedResponses (Computed number, that contains the number of reviewers, or more specifically, the exact number of elements in the “reviewers” field).
 - ReturnedResponses (This field is filled by the “(Child Post Back)” agent).
 - Field including parent information in order to allow communication.

Note These fields do not necessarily need to be on the form; they can also be dynamically added through the back end by the agent. If you use the sample code below, your form does not really need the fields.

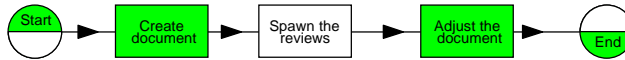
4. Create a multivalue field on the main form where the user must select some reviewers. Based on this multivalue field the child processes will be started and the document will be reviewed in the child process. Also create a response field where the results can be stored.
5. Enable your application for form-based initiation.

To enable your application for form-based initiation you should adjust the setup document and enable the option “Enable advanced initiation” in the “Initiation settings” section.

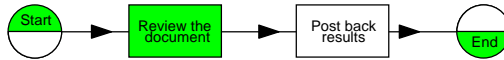
6. Create a process similar to the following example. In the automated activities you have to select the specified agents created in step one and two.

Note Of course you can extend these processes according to your needs. We oversimplified them in order to concentrate on the parent-child concept.

Parent process:



Child process:



Sample code for Parent Spawn agent

To “spawn” the child processes you need an automated activity which runs an automated agent. This agent should be based upon the “(OS Automation Template)”. Add the following LotusScript code in the CustomSubroutine part of the Automation template:

```
Function CustomSubroutine (binder As notesdocumentCollection,  
cover As notesdocument, main As notesdocument) As Integer
```

```
'This function is called for each Binder, that has to be  
processed by the agent
```

```
'Return 'True' to save the binder and route it to the  
succeeding activities: CustomSubroutine = True
```

```
'Return 'False' in case you want the binder to be processed by  
this agent again: CustomSubroutine = False
```

'The variable 'binder' contains a NotesDocumentCollection of all Binder Documents (including Cover and Main document)

'The variable 'cover' contains a NotesDocument, referencing to the Cover Document

'The variable 'main' contains a NotesDocument, referencing to the Main Document (after a join has occurred, the binder may contain several Main documents !)

```
Dim newdoc As NotesDocument
```

```
Dim rtitem As Notesrichtextitem
```

```
main.ExpectedResponses = 0
```

```
main.ReturnedResponses=0
```

```
Forall v In main.Reviewers
```

```
    main.ExpectedResponses=main.ExpectedResponses(0)+1
```

```
    Set newdoc = main.parentdatabase.createdocument
```

```
    ' create all items you need on child job main doc
```

```
    newdoc.form="Child Form"
```

```
    newdoc.CustomerName = main.CustomerName(0)
```

```
    newdoc.CustomerID=main.CustomerID(0)
```

```
    newdoc.Reviewer = v
```

```
    'fields below must be filled in order to start new job
```

```
    newdoc.NewProcessNameOS="(Child Process)"
```

```
    newdoc.NewJobNameOS=main.InstanceOS(0) & " - " & v
```

```
    newdoc.NewJobPriorityOS="High"
```

```
    newdoc.MailStatusOS="2"
```

```
    newdoc.InitiateOS="yes"
```

```
    'next field provides handle back from child job to parent binder
```

```
    newdoc.ParentFolderID=main.FolderIDOS(0)
```

```
    'next field provides handles to all parallel binders in parent job
```

```
    newdoc.ParentInstanceID=main.InstanceIDOS(0)
```

```

'add some doclinks to child job main document
    Set rtitem = New NotesRichTextItem( newdoc, "Body"
)

    Call rtitem.AppendText( "Cover: " )
    Call rtitem.AddTab( 2 )
    Call rtitem.AppendDocLink( cover, "Cover")
    Call rtitem.AddNewLine( 1 )
    Call rtitem.AppendText( "Main: " )
    Call rtitem.AddTab( 2 )
    Call rtitem.AppendDocLink( main, "Main" )

    Call newdoc.Save(False,True)
End Forall
CustomSubroutine=True

```

End Function

In the initialize section you must change the status the binder will get after the automated activity is finished. If you do this the parent binder will be "frozen" (won't be further routed without creating errors) because the backgrounder does not recognize this status. Therefore, change the row in the initialize script as follows:

```
doc.FolderStatusOS = "Ready to route"
```

Change to:

```
doc.FolderStatusOS = "Waiting for review responses"
```

This is important. Otherwise, your parent job will route to the next activity immediately. The code in the child agent will switch this value back to "Ready to route" if the expected number of responses is equal to the returned number of responses.

Sample code for Child Post Back Response

After the child job is finished the result of the child job should be added to the parent binder. Therefore, an automated activity is introduced in your process that runs an agent. The agent should be based on the "(OS Automation Template)." Place the following LotusScript code in the CustomSubroutine part of the agent:

```
Function CustomSubroutine (binder As notesdocumentCollection,
cover As notesdocument, main As notesdocument) As Integer

'This function is called for each Binder that has to be
processed by the agent

'Return 'True' to save the binder and route it to the
succeeding activities: CustomSubroutine = True

'Return 'False' in case you want the binder to be processed by
this agent again: CustomSubroutine = False

'The variable 'binder' contains a NotesDocumentCollection of
all Binder Documents (including Cover and Main document)

'The variable 'cover' contains a NotesDocument, referencing to
the Cover Document

'The variable 'main' contains a NotesDocument, referencing to
the Main Document (after a join has occurred, the binder may
contain several Main documents !)  
    Dim parentmain As NotesDocument
    Dim parentcover As NotesDocument
    Dim parentdoc As NotesDocument
    Dim parentbinder As NotesDocumentCollection
    Dim SearchParent As String
    Dim rt1, rt2 As Variant
    Dim dat2 As NotesDateTime
    Dim x As Integer
    Dim item As NotesItem
    Dim errorcode As Integer
    Dim success As Integer
```

```

' Find the parentbinder documents

    Set dat2 = New NotesDateTime("01/01/1900")

    SearchParent = | FolderIDOS = " | &
main.ParentFolderID(0) & |" |           'returns the
parentbinder of the document

    Set parentbinder =
main.ParentDatabase.Search(SearchParent, dat2, 0)

    If parentBinder.count <> 0 Then

        For x = 1 To parentbinder.count

            Set parentdoc =
parentbinder.GetNthDocument(x)

            If parentdoc.CoverDocOS(0) = "yes" Then Set
parentcover = parentdoc

            If parentdoc.MainDocOS(0) = "yes" Then Set
parentmain = parentdoc

            Next x

            Set item = parentmain.GetFirstItem ("ReviewResult")

            Call item.AppendToTextList(main.Reviewer(0) & ": "
& main.Decision(0))

'Add the child job comments to the parent maindoc

            Set rt1 = parentmain.GetFirstItem("Comments")

            Set rt2 = main.GetFirstItem("Comments")

            If ( rt1.Type = RICHTEXT And rt2.Type = RICHTEXT )
Then

                Call rt1.AddNewLine(1)

                Call rt1.AppendText(main.Reviewer(0) & ": " &
main.Decision(0))

                Call rt1.AddNewLine(1)

                Call rt1.Appendrtitem(rt2)

            End If

```

```

'increase number of returned responses on parentmain and check
whether this is last child job
'if I am the last then switch FolderStatusOS in parent binder
    parentmain.ReturnedResponses =
parentmain.ReturnedResponses(0) +1
    If parentmain.ExpectedResponses(0) =
parentmain.ReturnedResponses(0) Then

        For x = 1 To parentbinder.count
            Set parentdoc =
parentbinder.GetNthDocument(x)
            If Not (parentdoc Is Nothing) Then

parentdoc.FolderStatusOS = "Ready to route"

parentdoc.UserEditOS = "yes"

Call parentdoc.Save(False, True)
                End If
            Next x
        End If
'always save the parent main document to save the response
    Call parentmain.Save( False, True )
    Call parentcover.Save( False, True )

    CustomSubroutine = True
Else
'no Binder document was found
    Print "Unexpected error, no parentbinder found:
please check "
    End If
End Function

```

There are many situations where the parent-child concept can be used. It can also be customized to fit into specific situations. The most important thing when using this concept is to make certain that the status of the parent binder will always be correct. For example, scenarios where the parent binder also continues in the process can be risky!

It can easily happen that a parent binder waits for responses that will never come. It's also possible to let the parent binder be further routed, without waiting for the child processes, but the logic of such an approach has to be controlled by you and can lead to unexpected system behavior.

Extensions to Domino Workflow, such as parent-child processing, allow us to realize how complex the implementation of the whole workflow frame has to be in order to provide system stability and performance in various scenarios. Because of this complexity, they should never be used without serious consideration of all the consequences.

Appendix B

Advanced Web use of Domino Workflow

In Chapter 5 we discussed the basic Web features Domino Workflow offers as a standard user interface within the product. We also had a quick look at the advanced use of Domino Workflow on the Web. In this appendix we further describe how to use Domino Workflow on a Web client and how you are able to fit it in your intranet environment.

We first look at the questions you should ask yourself before starting to create the Web interface. After that we describe the programming you need to do to make this all work.

Before going through this appendix, it is useful for the reader to first download the example databases from the IBM Redbook Web site and walk through the design of the application database. It will make reading this chapter a lot easier. See the appendix “Additional Web material” for instruction on how to get the examples.

Before starting to build your Web user interface

There are some questions you should ask yourself before you start programming your Web user interface for your Domino Workflow application. The main three questions are listed below:

- **Who is your target audience?**

First you have to ask yourself how you want to design your application and think of what kind of audience will use your application. You can choose to make your application only accessible for Web clients or for both Web and Notes clients. You should also think about the type of functionality that may only be used by Web clients.

- **What look and feel do you want for your application?**

You should ask yourself if your application should fit in your company intranet environment. If it has to fit in your intranet environment, you should adapt the interface to the standards of your organization. If you do not need to standardize the interface for your intranet environment, you should define and document your own standards within the application.

- **How will you create your views?**

The “Desktop” views in your database are personal views, and personal views will not work on the Web. Therefore you should think of an alternative.

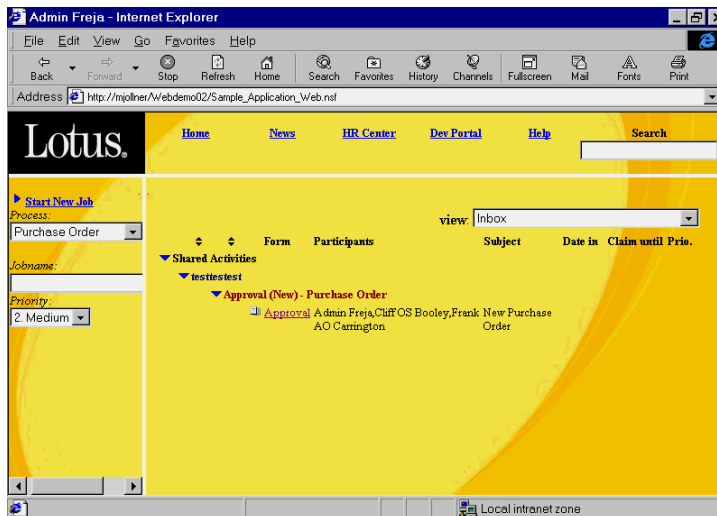
In the example we use throughout this appendix we have chosen to fit the application in the Lotus intranet environment. The application will only be used by Web clients. It would not be very difficult to make this application also available for Notes clients; however, to reduce the complexity in the sample we will focus only on Web clients.

The views are embedded in a frameset. The application is built for a Domino R5 Server and level 4 or higher browser. In the next section we describe how to program the solution.

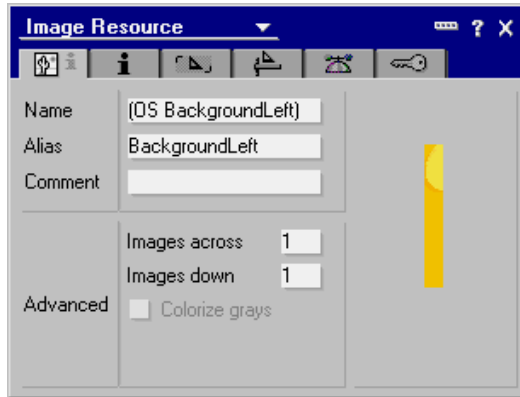
Programming the Web solution

All changes for the Web user interface are made in the application database. Many changes are very simple ones; however, some changes are more complicated. In the following list we describe the changes made to the application database. The simple ones will be explained first, and the more complicated ones will be discussed later in this section. (The sample database, which can be downloaded from the Redbooks Web site, includes the changes identified here):

1. The organization look is chosen. This example will describe the “The Lotus intranet look” as shown in the following figure.



2. The background pictures are imported as resource images in the design and used in the various forms.
 - a. (OS BackgroundHead): the image for forms for the TopFrame.
 - b. (OS BackgroundMain): the image for forms for the Mainframe.
 - c. (OS BackgroundLeft): the image for forms for the LeftFrame.



3. The intranet navigation is created for the TopFrame.

The intranet navigator is the form: (OS DWF Intranet Navigator). This form stores the links to other sites within your Intranet environment.

Note In the sample databases these links are not functional because most readers will not have access to the Lotus intranet environment. You can place your own intranet navigator here and make the links functional.
4. A navigation form is created for the LeftFrame: (OS DWF Workflow Navigator).

The (OS DWF Workflow Navigator) is the default navigator for the LeftFrame. In this form, functions are used to create a new job. These functions will be discussed later.
5. All views are adjusted: The cover document is not shown in the views.

In the views the selection formula is changed and the cover document will no longer be selected in the view. This is done by adding the following line to the view selection formula:

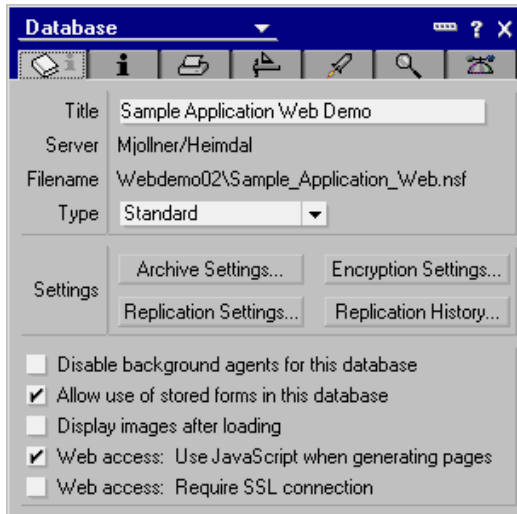
```
@UpperCase(CoverDocOS) != "YES"
```
6. Private views are changed.

Private views are changed to shared views because private views are not available on the Web. These views are embedded in a form called "(OS DWF Views)" where the option: "Show single category" is used, based on the value "@UserName". This solution will be discussed later in this chapter.

7. A frameset is created for use on the Web.

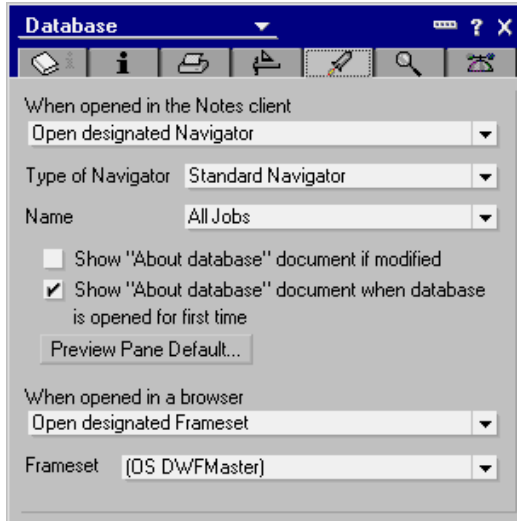
The frameset consists of three frames and is called the “(OS DWFMaster).” The TopFrame is the intranet navigator (form: “(OS DWF Intranet Navigator”), the LeftFrame default is the workflow navigator (form: “(OS DWF Workflow Navigator”) and the MainFrame is the default view form (form: “(OS DWF ViewDTInbox”).

8. In the general tab of the database properties the option “Web Access: Use JavaScript for generating pages” has to be enabled.



9. The options for Web launching must be set in the launch tab of the database properties.

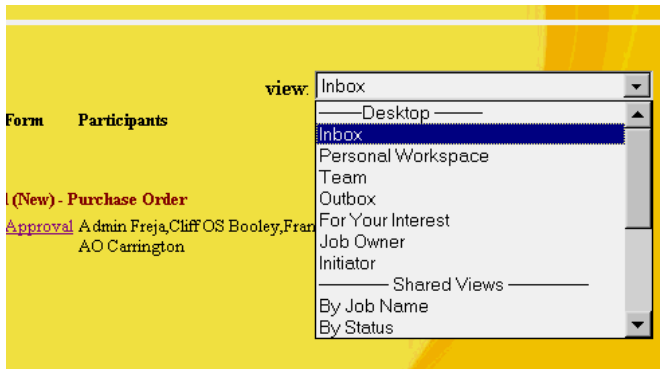
10. In the database properties the Launch options for the Web are changed. In this application we want the frameset "(OS DWFMaster)" to open when the application is opened, as shown in the following figure.



11. API and event design elements from the Domino Workflow 2.0 Developer's Toolkit are copied into the database so the events and the API can be used. For more information about these design elements, refer to the documentation in the Domino Workflow 2.0 Developer's Toolkit.

Views in the application

This application uses a view navigator that allows users to switch views by using a drop-down list in the upper left corner of the document (displayed in the main frame) as shown in the next figure. All views have been created using forms with embedded views.



In order to centralize all functionality, the subform “(OS DWF Views)” is being used. This subform needs two parameters that have to be set in the form that includes the subform:

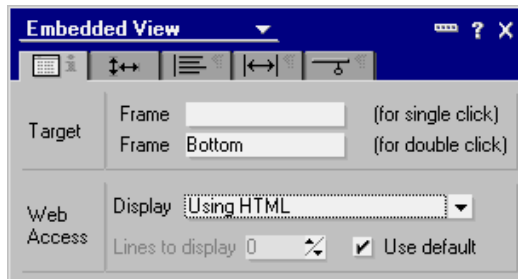
1. DWFSelctedView

This Field has to contain an identifier for the view name that is displayed by the subform. The identifier has to match an entry in the Selection formula of the embedded view in the subform. This formula finally determines the mapping to the View name.

2. DWFViewCategory

This Field has to contain the Category name that is to be displayed by the subform. Pass “*” if you want to display all categories of the view, pass the category name to display only values underneath this category.

The properties of the embedded view in the subform determine whether the view is displayed using HTML or the View applet. The general appearance can be changed by simply changing the properties of the embedded view on the subform.



If you want to display some views using the view applet and others using HTML you can choose the “Use View’s display properties” option and the view properties accordingly.

The first statement in the subform is:

```
<Meta http-equiv='Expires' content='0'>
```

It tells the Web browser to reload this page whenever the form is loaded. If a user navigates through Web pages using the back and forward browser options, this statement ensures that the view display is up to date. However, if you prefer a manual update of views via a refresh button, you can easily add such a button to the subform and remove the expired tag.

The view selection is based on JavaScript code, that is executed whenever the ComboBox field “DWFViewSelection” is changed (onChange event). To simplify the code, the script assumes that all “view form” names follow the syntax:

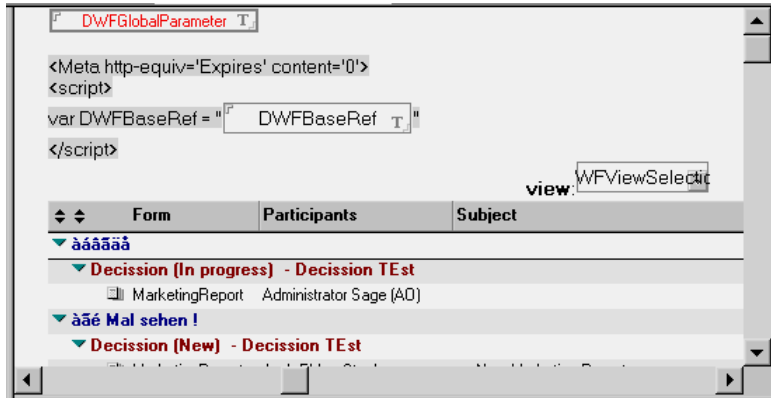
“DWFView”+ [Viewname] with the viewname being coded as the keyword value.

As another convention, all Desktop views (views that only display information that is relevant to the user) start with “DT”.

Note For programming convenience, viewnames should not contain any spaces. Categories within the combobox can be added by assigning their alias the value “noaction” — these keyword values are disregarded by the onChange procedure.

For each view a form has been created. This form only contains the two parameter fields (they are computed for display). For the Desktop views the value of the field DWFViewCategory is computed using the “@Username” formula. The background image of these forms determines the background images that are displayed in the user interface. In the demo, all “view form” background image properties are set to BackgroundMain.

The figure below shows the “(OS DWF Views)” subform.



The “Desktop” views

There are two types of views: “All Jobs” views and “Desktop” views:

- “All Jobs by ...” views are standard Notes shared views.
- “Desktop” views only display workflow binders that are related to the user that is logged in to the system. In the original Domino Workflow template, these views have been realized using private views for the Notes client (the default Web user interface uses lookups as a workaround, as private views are not available for Web clients). Since Domino R5 there is a new way to overcome private view limitations in general: a view can be embedded into another design element and programmed in a way that only entries under a definable category are displayed. This allows us to define shared views with the username as the first category entry. Then only those entries that appear under the usernames category are displayed.

Note Please keep in mind that these views become very large, depending on your workflow and the number of users involved. However, this approach eliminates the storage of private views on either the server or the Notes client — and makes otherwise private views available for Web clients.

There are several patterns that can be used to design Web views in the context of Domino Workflow: views can be displayed as HTML, or they can be displayed using the view applet. This can be set in your Embedded view properties.

HTML rendering provides a more Web-oriented style, whereas the view applet offers more functionality and a look and feel that is closer to the Notes Client.

Keep in mind, the approach to the cover document that is used in this database affects view design. In the original Domino Workflow template, users can see all binder documents including the cover document. The sequence in the view is cover document first, then main document and then all other documents. However, in this demo, the cover document is not presented as an explicit document, but its information is shown in the Left frame as a context to other binder documents. This implies two different view patterns that may seem to be contradictory. Either views can be constructed to only show cover documents, or views can be constructed that show all binder documents but not the cover document.

When hiding the binder documents, the user sees workflow activities in the views. To find out which documents are associated, the user has to open a workflow activity. The list of binder documents will then be displayed in the activity context frame. Therefore, the views have to be manipulated to open the main document instead of the cover document when clicked. As

the UniversalID of the main document is stored in the cover document, this can easily be achieved by inserting a new Column that displays a link to the main document:

```
[<A Href="\OSAllByFom/"+MainDocIDOS+"?Open\"target=_self>Open</A>]
```

This pattern cannot be used with the View Applet, as the applet always launches the original document.

When hiding the cover document, users see workflow binders in their views and can access a specific binder document directly. This approach is more binder centered than activity centered. In this case there is no need to offer links to other binder documents in the context frame as users can navigate to these document via views.

Change the standard views

No matter which pattern you prefer, the standard views have to be changed. Here are a few hints on how to modify the standard views:

1. Convert a private view to a shared view in which only those documents are displayed that appear under the usernames category.
 - Copy the Design of a private view into a shared view.
 - Change the selection formula: remove the restriction to only show documents that are related to @username.
 - Insert a new Column at the beginning of the view.

A new category has to be inserted categorizing all documents in which a given user fulfills a specific function that is relevant for the view. Example: In the Workspace view all workflow binders are displayed in which a user is the activity owner. Therefore a column has to be inserted that lists all entries of the field ActivityOwnerOS. The column sorting options "sort ascending," "categorized" and "show multiple entries as separate entries" have to be selected.

2. Hide the cover document or show only the cover document.

Depending on the display pattern that you want to use, you either need to include only cover documents or remove them from the list. Cover documents can be identified with the following selection formula: @UpperCase(CoverDocOS) = "YES".

3. Modify Column formulas.

Since a workflow binder may contain several documents, some information items that are valid for all binder documents, such as an Activity due date, are displayed only once per workflow binder. In the standard views, they are usually either displayed for the cover document or the main document.

Depending on whether you display the cover document or only the binder documents, you have to change the corresponding references:

If you are using the Activity pattern, change all references to MainDocOS = "yes" to CoverDocOS = "yes".

If you are displaying binder documents, change all references to CoverDocOS="yes" to MainDocOS="yes".

Some of the column values may need further attention because they display information that either is only available in binder documents (such as subject or document reservation) or information that only is available in the cover Document (such as ErrorTextOS).

4. Deciding on Categories.

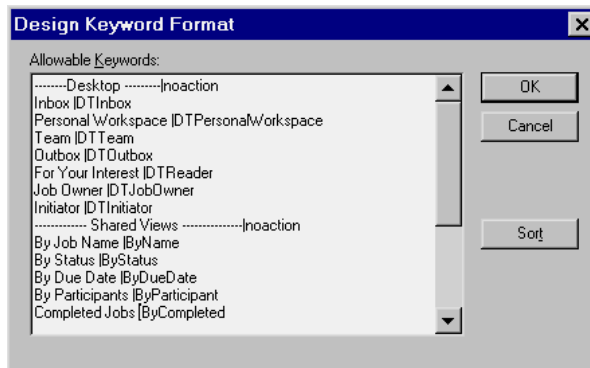
Depending on whether you want to use the View Applet or render views as HTML, you may or may not want to reduce the number of categories. This is a matter of personal preference or company standards.

Note Please keep in mind that view indexes require a significant amount of hard disk space. Delete all visible views that are not needed for your applications, especially the Desktop views which can become pretty large depending on your workflow and the number of users that are involved. Using the Activity pattern for view design will take less hard disk space than showing all binder documents.

How to add or remove views

To add a new view follow these steps:

1. Create a new view.
2. Open the "(OS DWF Views)" subform.
3. Add your view to the keyword list of the DWFSelctedView combobox field. Define a new keyword for the view; the keyword should not contain any spaces. If the view is a Desktop view, use "DT" as the first two letters of the keyword name, as shown in the following figure.



4. Add the mapping from the keyword to the real view name into the Selection Formula of the embedded view.
5. Copy one of the “(OS DWF View ...)” forms and rename it. Start the form name with “OS” to prevent the form from being displayed in the Domino Workflow Architect. You also have to add an alias name for the form. This alias name has to have the syntax “DWFView” + [view keyword name].
6. Change the Value formula of the field DWFViewSelection to the keyword that you have chosen for this view.

To remove views from the selection list simply do the following:

1. Remove the corresponding keyword entries from the field DWFSelctView in the subform (OS DWF Views).

The Workflow Navigator

The Workflow Navigator is displayed in the left frame. It makes up the workflow home navigator. The form used for this frame is “(OS DWF Workflow Navigator).” Its layout and content can easily be changed.

Starting new jobs

The functionality to start a new Workflow is inherited from the subform (OS DWF New Job) that has been included.

This subform offers several features:

1. JavaScript functionality to start a new Workflow Job.

The subform makes the function DWFNewJob(ProcessName, JobName, JobPriority, AdditionalParameters, Target) available.

This function initiates a new workflow and displays the main document of the first activity in the Target frame. Process Name, Job Name and Job Priority can be passed using the corresponding parameters. The AdditionalParameters variable may contain an array of strings. This array is then passed to the newly created job and made available within the field “DWFURLPARAMETER” in the cover document. If Target frame is “”, the default Target frame is being used.

The JavaScript function runs the DWFNewJob Agent using a URL to the agent. All parameters are passed via this URL. To handle spaces and special characters (such as double-byte characters) the URL string is encoded. The DWFNewJobAgent reads the URL parameters, decodes them, and extracts all the properties passed in to the agent. It then calls DWFNewJob function, which is available through the Domino Workflow Developer’s Toolkit API. Once the new job is initiated, a newly created

Event function QueryWebInitPrompt is called. You can use this event to modify the system behavior after a new job has been created, for example, to launch another document, or launch the new job in a new browser instance.

You can use this JavaScript function from any part of your own JavaScript or HTML code. You can, for example, create a button or a link that initiates a predefined process and associates a computed Jobname.

2. Optionally, fields that contain a list of startable processes and the default priorities can be generated.

If you want to create a user interface that displays all startable processes and presents a list of default priorities, you have to create a computed (for display) field with the name "DWFGetAvailableProcesses" and the value "yes." If this field value is set you can read the list of available processes from the field DWFAvailableProcesses, the list of priorities from the field DWFJobPriorities, and the default job priority from the field "DWFDefaultPriority."

3. JavaScript functionality to open a designated view.

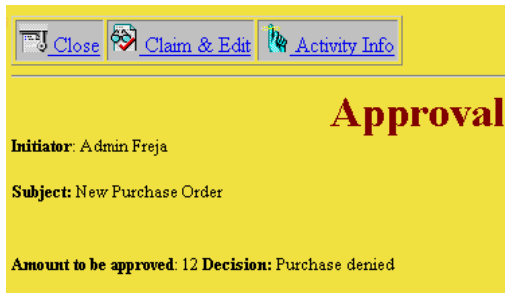
The JavaScript function DWFWorklistOpen(View, Target) opens the view whose keyword name has been passed to the function in the Target frame. Specify View = "" to launch the default view.

Binder documents / Domino Workflow Information Subform

All binder documents are displayed in the bottom frame. Instead of the standard Web subform that is shipped with Domino Workflow, this application uses two modified versions, each implementing a different style.

1. One style features action buttons similar to those in the Notes client, as shown in the following figure.

The Notes style subform is named "(OS Domino Workflow WEB)"; it replaces the standard subform for the Web client.



2. The other style displays actions in an alternative format, as shown in the next figure.

The Web style subform is named “(OS Domino Workflow WEB inactive).”



To switch between these designs, you can either change the subform formula on your form or change the subform names. Both subforms contain another subform: “(OS DWF Web functionality)”. This subform carries all the functionality shared between both styles. In a real-life scenario this subform should be resolved into the one used for display (for performance reasons).

Besides being a matter of personal taste, both styles have advantages and disadvantages.

- Displaying actions defined on form level.

The Notes styles are based on the Notes action bar feature. Actions inserted by the subform will automatically mix with actions owned by the form itself. Using the Web style, action buttons defined in the form will be displayed independently of the workflow action list.

- Using dynamic text for action titles

The text of Notes action buttons can only be defined at the time the form is designed. However, the Web style display allows you to change the text of actions according to the context. This allows you, for example, to display exclusive routing relations instead of the complete action (see Hide Complete Option).

There are a couple of additional features available which are not used on the standard subform shipped with the product:

- All information items that are required for a workflow activity are displayed in each binder document.

The Domino Workflow Architect allows you to define decisions, task lists, and exclusive and multiple routing relations. All of these features have in common, that users have to make a choice among different options before they can complete an activity. In the standard user interface, these options are displayed in the cover document, or are accessible from binder documents via actions. In this demo all these choices are automatically displayed in the binder documents. Before an Activity can be completed, all choices are checked using JavaScript. If users try to complete an activity

while any of the required choices have not been made, an error message is displayed right away and the completion is aborted.

- A JavaScript interface to all the functionality that is useful within a binder document.

All functionality that is useful within a binder document context, such as Claim, Complete, or Return Reassignment is provided via JavaScript functions. This allows you to design a binder document user interface that is completely detached from the original design shipped with the product.

This table lists all functions that are available in the application:

<i>Name</i>	<i>Parameter</i>	<i>Description</i>
DWFDocumentReload(Editmode)	Editmode = True: The document is reloaded in editmode. Editmode = False: The document is reloaded in reader mode.	This function reloads the current document without saving the document or doing any validation check. It may be used to load a document in edit mode even though it was opened in reader mode or vice versa.
DWFUtilGetUniqueString()	No parameter.	Returns a unique string. This utility function may be used to create a string that is appended to the URL to make sure that they are reloaded. Browsers may cache URLs, so when a URL is executed the second time, the result of the first query may be returned. By appending a unique string to each URL, the browser is forced to reload the URL.
DWFUtilGetFormFieldValue(Element)	Element: Reference to a radio button or checkbox input control.	Returns the values of all checked or selected elements as a string, divided by colons.
DWFActivityShowInfo()	No parameter.	Opens the Coverdocument in a new browser window. If you are not using another frame to display information about a workflow activity, this option allows users to gather information about their workflow context.

continued

<i>Name</i>	<i>Parameter</i>	<i>Description</i>
DWFDocumentSave(action)	Action = "complete" Saves the document and completes the activity. Action = "return" Saves the document and returns a reassignment. Action = "SaveAndReload" Saves the document and reloads it. Action = "save" Saves the document Action = "" Saves the document.	Saves the currently opened document. Depending on the specified action, the activity may also be completed or returned.
DWFActivityClaim(Documentid, Target)	Documentid: Documentid of the document from a binder that needs to be claimed and opened in the target frame. Target: Reference to the frame in which the claimed binder is opened.	Claims a binder and opens the document whose ID was passed in the target frame.
DWFNewJob(Processname, Jobname, JobPriority, AdditionalParameters, Target)	This function is described in the text above. See "Workflow Home"	

Other functionality

Adding your own code for activity validations

Sometimes it is useful to execute your own JavaScript evaluations before an Activity is completed. Even though validations can easily be added to Notes forms, it may often be useful to only execute them when a user clicks the Activity Completed button, and not every time a document is saved. You can add your own validation code by using this procedure:

1. Create a field: "DWFCustomJavaScriptEvents" on the form on which the "Activity Complete" action will take place.
2. Add "complete" as one of its list values.

Note If this field is set in the form that includes the "OS Domino workflow Information" or "OS Domino Workflow WEB" subform, before the include statement, each attempt to complete an Activity will

call a JavaScript function `DWFOnActivityCompletion()`. This function has to be defined on the Form level. If it returns a non empty string, an error will be prompted based on this string.

Adding custom Actions (Web style subform only)

You may want to add your own actions to the actionlist for a given activity. You can do so by including a field `DWFCustomActions` in your form design before including the subform. The text of this field (single value), will automatically be added to the action list. This allows you to add HTML code that generates custom actions such as a Cancel or Help option.

If you need to use the server URL, include “<DWFBBaseRef>” as a place holder.

If you want to use the same icon for this action that is used for the other action, include “<DWFActionIcon>” as a place holder.

Sample Value for the field `DWFCustomActions`:

```
"<BR>" + "<a href =\'JavaScript:YourFunctionCall(\\"a  
parameter\\")\' ID=\'DWFB\' >" + "<DWFActionIcon>" + "Action  
Text" + "</a>"
```

Summary

In this appendix we looked at ways of customizing your application for Web use with advanced Domino technology, JavaScript and the Domino Workflow Developer’s Toolkit. There are many things to customize when you want your application to be available for Web clients. Many of these are simple; however, some can be more challenging. This chapter arms the reader with the appropriate knowledge to not only create a working Web interface, but to design a powerful application.

Appendix C

LotusScript sample classes

To make it easier for you use LotusScript in your Domino Workflow application, the following design elements are provided in the Redbook LotusScript Classes database that is available for download from the IBM Redbooks Web site. See the appendix “Additional Web material” for instructions on how to get this database, which is in the file 5963ch12.zip.

<i>Design element type</i>	<i>Design element name</i>
LotusScript library	Redbook Binder
LotusScript library	Redbook Automation Agent (Deactivated Automation Jobs)
Agent	Redbook Automation Template
Agent	Redbook Activate Waiting Jobs

Copy these design elements to your Domino Workflow application database, along with the libraries of the Domino Workflow Developer’s Toolkit. See the documentation of the toolkit for a list of libraries.

To be able to use the RedbookBinder and RedbookBinderDocument classes in the **OS Application Events** library of the Domino Workflow toolkit, you should include a

Use “Redbook Binder”

statement in the options section of the **OS Application Events** library.

Redbook binder library

This library contains two classes:

- RedbookBinder
- RedbookBinderDocument

RedbookBinder Class

Activity.id As String (read only)

Refers to the ActivityIDOS field on the cover document.

Activity.Owner As String (read only)

Refers to the ActivityOwnerOS field on the cover document.

Activity.Name As String (read only)

Refers to the ActivityOS field on the cover document.

Activity.Description As String (read only)

Refers to the ActivityDescriptionOS field on the cover document.

Activity.Claim(OnBehalfOf As String, ErrorCode As Integer) As Integer

Note This function maps directly to the Domino Workflow Developer's Toolkit DWFActivityClaim. Refer to the Domino Workflow Developer's Toolkit for the description of the parameters.

Activity.Complete(ImmediateRouting As Integer, FailureList As Variant, ErrorMessage As String, ErrorCode As Integer) As Integer

Note This function maps directly to the Domino Workflow Developer's Toolkit DWFActivityComplete. Refer to the Domino Workflow Developer's Toolkit for the description of the parameters.

Activity.Reassign(ParticipantList As Variant, NotifyByEmail As Integer, ReturnReassignment As Integer, DoNotAllowFurtherReassignment As Integer, NewActivityName As String, NewActivityDescription As String, ErrorMessage As String, ErrorCode As Integer) As Integer

Note This function maps directly to the Domino Workflow Developer's Toolkit DWFActivityReassign. Refer to the Domino Workflow Developer's Toolkit for the description of the parameters.

Job.id As String (read only)

Refers to the InstanceIdOS field on the cover document.

Job.Owner As Variant (read only)

Refers to the JobOwnerOS field on the cover document.

Job.Name As String (read only)

Refers to the InstanceOS field on the cover document.

Job.Initiator As String (read only)

Refers to the InitiatorOS field on the cover document.

Automation.Wait.Remove

Removes the AutomationWaitTill and AutomationWaitInterval fields from the cover (used by the RedbookAutomationAgent class after a successful run of an agent).

Automation.Wait.Till As Variant

The reactivation date for failed automated activities. Refers to the AutomationWaitTill field on the cover document. Defaults to Now minus 1 hour if not available.

Automation.Wait.Interval As Double

The current wait interval for a failed automation activity. Refers to the AutomationWaitInterval field on the cover document. Defaults to 1 if not available.

Automation.Wait.Failed(increment As Integer, maximum As Integer)

Set the new activation date and interval. Increase the last interval with “increment” percent up to “maximum” hours.

Automation.Remove

Remove the AutomationTypeOS and AutomationNameOs fields from the cover document, after a successful run of an automation agent.

Automation.Name As String (read only)

The name of the agent that processes this binder. Refers to the AutomationNameOS field on the cover document.

Automation.Type As String (read only)

Contains “Agent” in case of automation activity with LotusScript agents. Refers to the AutomationTypeOS field on the cover document.

Automation.Activate

If the binder status begins with an asterisk (*) it is removed. Used to reactivate waiting automation jobs. Used by the “Redbook Activate Waiting Jobs” agent.

Automation.Deactivate

If the Binder status does not begin with an asterisk (*) it is prefixed to the status. Used by the “RedbookAutomationAgent” class.

Process.id As String (read only)

Refers to the ProcessIDOS field. on the cover document.

Process.Name As String (read only)

Refers the ProcessOS field on the cover document.

Documents.List As Variant (list)

Produces a List variable containing all the NotesDocuments of this binder including the cover and main document.

Documents.Add(Doc As NotesDocument, ErrorCode As Integer) As Integer

Note This function maps directly to the Domino Workflow Developer's Toolkit DWFBinderAddDocument. Refer to the Domino Workflow Developer's Toolkit for the description of the parameters.

Documents.Remove(doc As Notesdocument, DeleteAllDWFFields As Integer, ErrorCode As Integer) As Integer

Note This function maps directly to the Domino Workflow Developer's Toolkit DWFBinderRemoveDocument. Refer to the Domino Workflow Developer's Toolkit for the description of the parameters.

Main As RedbookBinderDocument (read only)

The main document of the binder. In case there is more then one, the first one is returned.

Cover As RedbookBinderDocument (read only)

The cover document of the binder.

Save(Byval force As Integer)

Save all the documents of the binder.

ParentDatabase As NotesDatabase (read only)

The database where this binder comes from. Refers to the cover document.

Status As String

The status of the binder. Refers when reading to the FolderStatusOS field on the cover and will write to the FolderStatusOS field on all documents of the binder.

Id As String (read only)

Refers to the FolderIdOS field on the cover document. Binders can have the same Binder.Job.Id and different Binder.Id if binders are split due to parallel routing.

Reroute(DestinationActivityId As String, ErrorMessage As String, ErrorCode As Integer) As Integer

Note This function maps directly to the Domino Workflow Developer's Toolkit DWFBinderReroute. Refer to the Domino Workflow Developer's Toolkit for the description of the parameters.

new(BinderKey As Variant)

Binderkey can be any NotesDocument from a binder or the ID of the binder (FolderIdOS).

RedbookBinderDocument Class

Note As NotesDocument (read only)

The Notesdocument from which this object reads its data.

RemoveOSItem(OSItemName As String)

Removes specified item after appending the string "OS".

LastUserEdit As Variant

Refers to the UserEditOS field of the document. It should be set by an automated activity for all documents it changes.

Status As String (read only)

Refers to the DocStatusOS field of the document.

Id As String (read only)

Refers to the SourceIdOS field of the document.

isCover As Integer (read only)

Is true if this is the cover document of the binder.

isMain As Integer (read only)

Is true if the this a main document of the binder. Before a join of parallel routes, it can have more then one main document.

new(note As Variant)

The parameter Note should be a NotesDocument object and part of a Domino Workflow binder.

Redbook automation agent library

This library contains one class which you can use as the base class for your own automated activity agents (see also the Redbook automation template agent later in this section).

RedbookAutomationAgent

session As NotesSession

An instance of NotesSession.

Agent As NotesAgent

An instance of NotesAgent referring to the current agent.

BinderCol As NotesDocumentCollection

An instance of NotesDocumentCollection containing all the Cover documents this agent has to process.

Binder As RedbookBinder

An instance of RedbookBinder containing a reference to the currently processed binder.

CustomSubroutine() As Integer

The routine which has to be overridden by the sub class. If not overridden, an error will occur. This routine is called by Run.

Run(increment As Integer, Interval As Integer)

Called by the initialize routine. Increment is the percentage with which the wait interval increases each time the agent fails. The interval won't become larger than the specified maximum. If a maximum of 0 is specified the agent will not be deactivated if it fails

Deactivated automation jobs view

This is a hidden view that selects all cover documents of jobs which have been deactivated through prefixing an asterisk (*) to the binder status. All documents are sorted on the AutomationWaitTil date field, the oldest date on top. The Redbook Activate Waiting Jobs agent will use this view to activate jobs whose AutomationWaitTil date has passed.

Redbook automation template agent

This agent is an example of how to use the RedbookAutomationAgent class to create your own automated activity agents. Create a copy and add your own code to the CustomSubroutine function of the class.

Redbook activate waiting jobs agent

Jobs whose automated activity agents return “false” in the custom subroutine can be deactivated depending on the parameters of the agent. When that happens the Domino Workflow background agent is not able to see them anymore, and consequently won’t spend any time on them. This agent is used to activate waiting jobs, to give the Domino Workflow background agent a chance to process them again.

Special notices

This publication is intended to help you create customized solutions with Lotus Domino Workflow 2.0.

The information in this publication is not intended as the specification of any programming interfaces that are provided by Lotus Domino. See the publications section of the announcement for Lotus Domino and related products for more information about what publications are considered to be product documentation.

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM products, programs, or services may be used. Any functionally equivalent program that does not infringe on any IBM intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 USA.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM Corporation, Dept. 600A, Mail Drop 1329, Somers, NY 10589 USA.

Such information may be available subject to appropriate terms and conditions, including, in some cases, payment of a fee.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The information about non-IBM ("vendor") products in this manual has been supplied by the vendors, and IBM assumes no responsibility for its accuracy or completeness. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate

and integrate them into the customer's operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

Any pointers in this publication to external Web sites are provided for convenience only and do not in any manner serve as an endorsement of these Web sites.

Any performance data contained in this document was determined in a controlled environment, and therefore the results that may be obtained in other operating environments may vary significantly. Users of this document should verify the applicable data for their specific environment.

This document contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples contain the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

Reference to PTF numbers that have not been released through the normal distribution process does not imply general availability. The purpose of including these reference numbers is to alert IBM customers to specific information relative to the implementation of the PTF, when it becomes available to each customer according to the normal IBM PTF distribution process.

The following terms are trademarks of the International Business Machines Corporation in the United States and/or other countries:

AIX
AS/400
DB2
IBM®
MQSeries
OS/2
OS/Warp

The following are trademarks of Lotus Development Corporation in the United States and/or other countries:

Domino Workflow
Domino.Doc
LotusScript®
Lotus SmartSuite®
Notes Mail®
NotesPump
NotesSQL

Lotus®
Lotus Domino
Lotus Notes®
RealTime Notes
SmartIcons®

The following terms are trademarks of other companies:

Tivoli, Manage. Anything. Anywhere., The Power to Manage., Anything. Anywhere., TME, NetView, Cross-Site, Tivoli Ready, Tivoli Certified, Planet Tivoli, and Tivoli Enterprise are trademarks or registered trademarks of Tivoli Systems Inc., an IBM company, in the United States, other countries, or both. In Denmark, Tivoli is a trademark licensed from Kjøbenhavn Sommer - Tivoli A/S.

C-bus is a trademark of Corollary, Inc.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and/or other countries.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States and/or other countries.

PC Direct is a trademark of Ziff Communications Company and is used by IBM Corporation under license.

ActionMedia, LANDesk, MMX, Pentium and ProShare are trademarks of Intel Corporation in the United States and/or other countries.

UNIX is a registered trademark in the United States and other countries licensed exclusively through the Open Group.

SET, SET Secure Electronic Transaction and the SET logo are trademarks owned by SET Secure Electronic Transaction LLC.

Other company, product or service names may be the trademarks or service marks of others.

Additional Web material

Additional Web material is referenced in this book and can be found on the IBM Redbooks Web site. The material is:

<i>File name</i>	<i>Description</i>
5963-ch4.zip	Four databases with the Domino Workflow setup for the scenarios in Chapter 4 “Defining your first process.” The Zip file includes a read.me file that explains how to set up the databases for being viewed in the Domino Workflow Architect.
5963-ch6.zip	Four databases with the Domino Workflow setup for the scenario where Domino Workflow functions have been added to an existing application. If you simply want to see which design elements have been added to the existing application, take a look at the database named dlibapp.nsf. The Zip file includes the template for the original application. Note You must be using Notes R5.0 or higher to work with these databases.
5963ch12.zip	Database with script libraries and other design elements that shows an example of encapsulating Domino Workflow properties using an object-oriented approach.
5963appb.zip	Four databases with the Domino Workflow setup for an example of how to write advanced Web user interfaces. The demo highlights different patterns for modifying the Domino Workflow user interface to match specific requirements. It also contains building blocks that can be reused and further customized to suit special needs. Part of this demo is described in Appendix B. Note You must be using Notes R5.0 or higher to work with these databases.

How to get the Web material

To get the Web material point your Web browser to:

ftp://www.redbooks.ibm.com/redbooks/SG245963

Alternatively, you can go to the redbooks Web site at:

http://www.redbooks.ibm.com

Select Additional Materials and open the file that corresponds with the redbook form number.

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

IBM Redbooks

For information on ordering these publications see, “How To Get IBM Redbooks.”

- *Lotus Domino 5.0: A Developers Handbook*, IBM form number SG24-5331, Lotus part number CC7EDNA
- *Performance Considerations for Domino Applications*, IBM form number SG24-5602, Lotus part number CT7V6NA
- *COM Together — with Domino*, IBM form number SG24-5670, Lotus part number CT7V7NA
- *LotusScript for Visual Basic Programmers*, IBM form number SG24-4856, Lotus part number 12498
- *Developing e-business Applications Using Lotus Enterprise Solution Builder R3.0*, IBM form number SG24-5405, Lotus part number CT69TNA
- *Building a Portal with Lotus Domino R5*, IBM Redpaper REDP0019 (only available in softcopy from the IBM Redbooks Web site)
- *Developing Web Applications Using Lotus Notes Designer for Domino 4.6*, IBM form number SG24-2183, Lotus part number 12974
- *Lotus Notes 4.5: A Developers Handbook*, IBM form number SG24-4876, Lotus part number AA0425
- *Creating Customized Solutions with Domino.Doc*, IBM form number SG24-5658, Lotus part number CC6X3NA
- *Using VisualAge for Java to Develop Domino Applications*, IBM form number SG24-5424, Lotus part number CT6ENNA
- *Connecting Domino to the Enterprise Using Java*, IBM form number SG24-5425, Lotus part number CT6EMNA
- *Lotus Domino for AS/400: Integration with Enterprise Applications*, IBM form number SG24-5345, Lotus part number CT7BMNA

- *Lotus Solutions for the Enterprise, Volume 1. Lotus Notes: An Enterprise Application Platform*, IBM form number SG24-4837, Lotus part number 12968
- *Lotus Solutions for the Enterprise, Volume 2. Using DB2 in a Domino Environment*, IBM form number SG24-4918, Lotus part number CT69BNA
- *Lotus Solutions for the Enterprise, Volume 3. Using the IBM CICS Gateway for Lotus Notes*, IBM form number SG24-4512
- *Lotus Solutions for the Enterprise, Volume 4. Lotus Notes and the MQSeries Enterprise Integrator*, IBM form number SG24-2217, Lotus part number 12992
- *Lotus Solutions for the Enterprise, Volume 5. NotesPump, the Enterprise Data Mover*, IBM form number SG24-5255, Lotus part number CT69DNA
- *Enterprise Integration with Domino for S/390*, IBM form number SG24-5150

Other Lotus-related IBM Redbooks

The publications listed in this section may also be of interest:

- *Lotus Notes and Domino Take Center Stage: Upgrading from R4 to R5*, IBM form number SG24-5630, Lotus part number CT6FDNA
- *A Roadmap for Deploying Domino in the Organization*, IBM form number SG24-5617, Lotus part number CT6P8NA
- *The Three Steps to Super.Human.Software: Compare, Coexist, Migrate; From Microsoft Exchange to Lotus Domino, Part One: Comparison*, IBM form number SG24-5614, Lotus part number CT7QTNA
- *The Three Steps to Super.Human.Software: Compare, Coexist, Migrate; From Microsoft Exchange to Lotus Domino, Part Two: Coexistence and Migration*, IBM form number SG24-5615, Lotus part number CT7QWNA
- *Lotus Notes and Domino R5.0 Security Infrastructure Revealed*, IBM form number SG24-5341, Lotus part number CT6TPNA
- *The Next Generation in Messaging. Moving from Microsoft Mail to Lotus Notes and Domino*, IBM form number SG24-5152, Lotus part number CT7SBNA
- *Eight Steps to a Successful Messaging Migration: A Planning Guide for Migrating to Lotus Notes and Domino*, IBM form number SG24-5335, Lotus part number CT6HINA

- *Deploying Domino in an S/390 Environment*, IBM form number SG24-2182, Lotus part number 12957
- *The Next Step in Messaging: Upgrade Case Studies for Lotus cc:Mail to Lotus Domino and Lotus Notes*, IBM form number SG24-5100, Lotus part number 12992
- *The Next Generation in Messaging. Moving from Novell GroupWise to Lotus Notes and Domino*, IBM form number SG24-5321, Lotus part number CT7NNNA
- *High Availability and Scalability with Domino Clustering and Partitioning on Windows NT*, IBM form number SG24-5141, Lotus part number CT6XMIE
- *From Client/Server to Network Computing, A Migration to Domino*, IBM form number SG24-5087, Lotus part number CT699NA
- *Netfinity and Domino R5.0 Integration Guide*, IBM form number SG24-5313, Lotus part number CT7BKNA
- *Lotus Domino R5 for IBM RS/6000*, IBM form number SG24-5138, Lotus part number CT7BHNA
- *Lotus Domino Release 4.6 on IBM RS/6000: Installation, Customization and Administration*, IBM form number SG24-4694, Lotus part number 12969
- *High Availability and Scalability with Domino Clustering and Partitioning on AIX*, IBM form number SG24-5163, Lotus part number CT7J0NA
- *Lotus Domino for S/390 Release 4.6: Installation, Customization & Administration*, IBM form number SG24-2083
- *Lotus Domino for S/390 Performance Tuning and Capacity Planning*, IBM form number SG24-5149, Lotus part number CT6XNIE
- *Porting C Applications to Lotus Domino on S/390*, IBM form number SG24-2092, Lotus part number
- *Managing Domino/Notes with Tivoli Manager for Domino, Enterprise Edition, Version 1.5*, IBM form number SG24-2104

IBM Redbooks collections on CD-ROM

Redbooks are also available on the following CD-ROMs. Click the CD-ROMs button at <http://www.redbooks.ibm.com/> for information about all the CD-ROMs offered, updates and formats.

<i>CD-ROM Title</i>	<i>Collection Kit Number</i>
Lotus Redbooks Collection	SK2T-8039
Tivoli Redbooks Collection	SK2T-8044
Application Development Redbooks Collection	SK2T-8037
Netfinity Hardware and Software Redbooks Collection	SK2T-8046
RS/6000 Redbooks Collection (BkMgr Format)	SK2T-8040
RS/6000 Redbooks Collection (PDF Format)	SK2T-8043
AS/400 Redbooks Collection	SK2T-2849
Transaction Processing and Data Management Redbooks Collection	SK2T-8038
Networking and Systems Management Redbooks Collection	SK2T-6022
System/390 Redbooks Collection	SK2T-2177
IBM Enterprise Storage and Systems Management Solutions	SK3T-3694

Other resources

These publications are also relevant as further information sources:

- *The Lotus Notes & Domino 5.0 Application Development Best Practices Guide*, Lotus documentation, available online at <http://www.lotus.com/bpg>
- *Inside Notes: Domino Architecture Documentation*, Lotus documentation, available online first quarter 2000 at <http://notes.net/notesua.nsf/find/inside-notes>
- *Lotus Knowledge Base* - contains Tech Notes and Papers, available online at <http://support.lotus.com/>

Web sites

These Web sites are also relevant as further information sources:

- <http://www.lotus.com/workflow>
Home page for Domino Workflow - contains product information , white papers and a discussion forum.
- <http://www.lotus.com/developer>
Lotus Developer Network - Lotus' primary destination for the latest developer information and resources. Contains articles about new and current technologies along with relevant tips and techniques to help you build dynamic collaborative e-business applications.
- <http://notes.net/>
Notes.net from Iris — the developers of Notes and Domino — is a technical Web site with discussion forums, documentation and the Webzine Iris Today with many good articles about technical details of Domino.
- <http://www.ibm.com/developer/>
The IBM developerWorks Web site is designed for software developers, and features links to a host of developer tools, resources, and programs.
- <http://support.lotus.com/>
Lotus Support's Web site - Search using keywords or browse the Lotus Knowledge Base and locate helpful and informative tech notes and technical papers for the entire Lotus Product family. This source of information contains the latest technical information updated hourly.

How to get IBM Redbooks

This section explains how both customers and IBM employees can find out about IBM Redbooks, redpieces, and CD-ROMs. A form for ordering books and CD-ROMs by fax or e-mail is also provided.

- **Redbooks Web Site** <http://www.redbooks.ibm.com>

Search for, view, download or order hardcopy/CD-ROM Redbooks from the IBM Redbooks Web site. Also read redpieces and download additional materials (code samples or diskette/CD-ROM images) from this Redbooks site.

Redpieces are redbooks in progress; not all redpieces become redbooks and sometimes just a few chapters will be published this way. The intent is to get the information out much quicker than the formal publishing process allows.

- **E-mail Orders**

Send orders by e-mail including information from the IBM Redbooks fax order form to:

e-mail address

In United States: usib6fpl@ibmmail.com

Outside North America: Contact information is in the "How to Order" section at this site:
<http://www.elink.ibm.com/pbl/pbl/>

- **Telephone Orders**

United States (toll free) 1-800-879-2755

Canada (toll free) 1-800-IBM-4YOU

Outside North America Country coordinator phone number is in the "How to Order" section at this site:

<http://www.elink.ibm.com/pbl/pbl/>

- **Fax Orders**

United States (toll free) 1-800-445-9269

Canada (toll free) 1-800-267-4455

Outside North America Fax phone number is in the "How to Order" section at this site:

<http://www.elink.ibm.com/pbl/pbl/>

The latest information for customers may be found at the IBM Redbooks Web site.

IBM intranet for employees

IBM employees may register for information on workshops, residencies, and redbooks by accessing the IBM intranet Web site at <http://w3.itso.ibm.com/> and clicking the ITSO Mailing List button. Look in the Materials repository for workshops, presentations, papers, and Web pages developed and written by the ITSO technical professionals; click the Additional Materials button. Employees may access **MyNews** at <http://w3.ibm.com/> for redbook, residency, and workshop announcements.

IBM Redbooks fax order form

Please send me the following:

Title	Order Number	Quantity
_____	_____	_____
_____	_____	_____
_____	_____	_____
_____	_____	_____
_____	_____	_____
_____	_____	_____
_____	_____	_____
_____	_____	_____

First name _____ Last name _____

Company _____

Address _____

City _____ Postal code _____ Country _____

Telephone number _____ Telefax number _____ VAT number _____

Invoice to customer number _____

Credit card number _____

Credit card expiration date _____ Card issued to _____ Signature _____

We accept American Express, Diners, Eurocard, MasterCard, and Visa. Payment by credit card not available in all countries. Signature mandatory for credit card payment.

Index

Symbol

1:N relations. *See* Parent-child processing

A

Access Control List, 55

Activity

- attributes, 154
- automated, 17, 250
- automated - scenarios
 - using, 174
- create on the fly, 223
- creating, 81
- definition, 16
- owner - change team, 219
- owner - reassign, 223
- owners, 20
- parallel, 24
- potential owners, 20
- reassign, 223
- working on, 35

Activity owner

- multiple potential, 203

Activity properties, 82

- description, 85
- forms, 84
- join settings, 126
- owner, 83
- reassign, 85
- tasks, 84
- timing, 85

Agents

- enabling, 49
- scheduling on multiple servers, 202

Alternative organization

- directory, 191
- advantages, 181
- changing your data sources, 197
- custom integration, 198
- disadvantages, 182
- Domino Directory, 191
- mapping files, 192
- replacing design elements, 195

- resource database, 193
 - specifying, 193
- Application database, 26
 - administration views, 28
 - configuring, 43, 53, 245
 - customizing, 157
 - multiple, 39
 - replicas, 39
 - setup, 38
 - views, 27
- Approval cycles, 110
- Architect
 - creating a database profile, 75
 - creating a new process, 77
 - description, 19, 31
 - interface, 33, 151
 - settings, 32
 - storage facility, 33
- Archive
 - database, 30, 189
 - scenarios, 239
 - setup, 237
 - strategies, 237
- Audit Trail, 189, 235
 - database, 31
 - view, 232
- Automated activity, 265
 - (de)activating binders, 260
 - adding agent, 254
 - body formula, 125
 - creating agent, 254
 - CustomSubroutine in agent, 254
 - failing agents, 259
 - how agents work, 255
 - Notes UI objects in agents, 262
 - run LotusScript agent, 253
 - run server program, 253
 - Send Mail, 121, 253
 - send to formula, 122
 - subject formula, 124
- Automation agent class
 - run method parameters, 260

B

Backgrounder, 231

Binder, 16

- attributes, 154
- cover, 16
- documents, 16
- join, 25
- main document, 16
- multiple copies, 24
- structure, 16

Business object library, 33, 151

Business process

- classification, 4
- cooperative, 5
- managing, 9
- mixed forms, 6
- semi-structured, 5
- structured, 5
- transparency, 8
- visualizing, 8, 19, 36

C

Cache, 231, 248

Calendar document, 70

Change management, 9

Claim

- automatic, 175

Cluster, 157

Custom attributes, 100, 149, 154

- defining, 101
- using in application forms, 102

Customization

- toolkit, 157

D

Department documents

- alternative way to add members, 70
- creating, 65
- manager of a department, 66
- surrogates, 67

Design repository, 33, 152

- Design repository database
 - configuring, 43
- Developer's toolkit
 - events library, 261
- Distributed processing, 201
- Domino Directory, 41, 134, 196
- Domino server requirements, 44
 - access rights, 44
 - unrestricted agents, 49
- Domino Workflow
 - adding to an existing
 - application, 141
 - application, 26
 - Architect, 31
 - architecture, 15, 25
 - archive, 30
 - audit trail, 31
 - design repository, 33
 - developer's toolkit, 256, 261
 - engine, 25
 - events, 261
 - integration into
 - Domino/Notes, 13
 - integration with other systems, 13
 - main characteristics, 11, 13
 - organization, 29
 - process definition, 30
 - running, 219
 - Toolkit, 157
 - Viewer, 233
- Domino Workflow elements, 15
 - activities, 16
 - activity owners, 16
 - application database, 26
 - Architect, 19, 31
 - archive, 30
 - audit trail, 31
 - binder, 16
 - design repository, 33
 - engine, 25
 - job, 21
 - job owners, 22
 - organization directory, 29
 - potential activity owners, 20
 - process definition, 21, 30
 - routing relation, 18
 - team, 24
- Domino.Doc integration
 - archiving, 209, 211
 - automated search agents, 212
 - automatic search, 209
 - check in/check out, 209, 213
 - initiation, 209, 211

E

- Enabling new documents, 107
- Engine
 - application, 26
 - customizing, 157
 - elements, 25
 - organization, 29
- Error handling
 - in the run-time, 22
- Events, 158
 - initiating a job, 168
- Events library
 - modifying, 261
 - NotesUIWorkspace, 262

F

- Form-based initiation, 266
- Forms
 - implementing for application, 71
- Formula
 - body formula, 125
 - owner, 118
 - send to formula, 122
 - subject formula, 124
- Frameset, 139

I

- Initiate jobs
 - advanced - settings, 160
 - form-based, 162
 - mail-based, 161
 - manually, 159
 - scenario, 165
 - troubleshooting, 159, 162
 - view, 230
- Initiation settings, 133
- Integration, 209. *See Also*
 - Domino.Doc integration
 - Domino.Doc XML, 215

J

- JavaScript, 140, 281
 - opening designated navigator, 286
 - starting new jobs, 285
 - validations, 289
- Job
 - change on the fly, 225
 - complete, 36

- definition, 21
- information, 235
- initiation methods, 34, 158
- initiate, 34, 158, 248
- owner, 22, 249
- owner - change, 250
- owner - reroute, 225
- participants, 20, 22, 24
- view, 36, 229, 233
- Job Properties, 29, 186
 - for accessing fields, 143
 - activity owner, 118
 - predefined, 187
- Joins, 99
 - concept, 25
 - disabling, 25, 99

L

- Loops
 - to same activity, 175
- LotusScript
 - agent for automated activities, 253
 - automation agent class, 257
 - binder document class, 255, 256
 - developer's toolkit, 256
 - library name, 257
- LotusScript agent
 - use in the automated activity, 171, 250
 - develop, 171
 - schedule, 246

M

- Mapping files
 - adding your own OrgaTypes, 194
- Mobile users, 203
- Multi server
 - cluster, 203
 - different locations, 205
 - scheduling agents on, 202
- Multiple databases, 201
 - replicas, 202

N

- Notes client, 130
- Notes UI objects
 - in scheduled agents, 262
- Notifications
 - getting too early, 203

O

- Organization directory, 179
 - advanced use, 182
 - alternative, 41
 - configuring, 43, 50
 - copying and pasting from
 - Domino Directory, 57
 - database, 29
 - disadvantages, 181
 - elements, 29, 249
 - importing from Domino
 - Directory, 56
 - job properties, 186
 - Out of office, 180
 - relations, 183
 - resources, 37, 188
 - standard relations, 183

P

- Parent-child processes, 176, 265
 - child process, 267
 - parent process, 267
- Performance
 - archiving, 205
- Predefined job properties, 88
 - Initiator, 88
- Process
 - activating, 90, 247
 - archive, 30
 - attributes, 155
 - cache, 133, 248
 - changes on the fly, 219
 - checking syntax, 90
 - creating from a copy, 94
 - definition, 21, 30, 31
 - design, 31, 153
 - hierarchy, 155, 176
 - history, 31
 - map, 19
 - participants, 20, 22, 24
 - preparing for process design, 61
 - reuse, 152
 - versions, 30
 - view, 233
- Process definition database
 - configuring, 43
- Process properties, 77
 - description, 81
 - forms, 80
 - initiators, 79
 - job owner, 78
 - timing, 80

Processes

- Sub, 202
- Prototyping strategies, 91

R

- Reassign, 223
- Relations, 29, 183
- Replication
 - all databases available, 202
 - conflicts reported, 202, 207
 - local, 203
 - selective, 206
- Reroute, 225
- Resource database. *See* Alternative
 - organization directory
- Resources, 188
 - application, 189
 - mail addresses, 189
 - server driven programs, 190
- Role documents
 - alternative way to add
 - members, 70
 - creating, 70
- Routing
 - problems, 249
 - strategies, 35
- Routing relations, 98
 - by decision, 112
 - conditional, 105
 - definition, 18
 - Exclusive Choice, 98
 - problems, 249
 - various kinds, 19

S

- Save conflicts, 251
- Send mail, 170
 - mail inquiry, 222
- Server
 - cluster, 203
 - reserved for scheduled
 - agents, 205
- Server program
 - execute, 170
 - set parameters, 171
- Setup document, 132
- Subprocess
 - definition, 155
 - properties, 157
 - settings, 156
 - using, 156

T

- Team
 - change, 219
 - concept, 24
 - protecting documents, 204
- Toolkit, 157
- Troubleshooting, 247
 - views, 230

V

- Viewer, 233
- Views, 229
 - administration, 230, 231, 232
 - all documents, 231
 - audit trail, 232
 - cache, 231
 - initiation, 230
 - tests, 231
 - troubleshooting, 230

W

- Web browser, 130
- Web interface
 - customizing, 139
- Web user interface, 275
 - converting private views, 283
 - database properties, 278
 - embedded view, 280, 282
 - framesetting, 278
 - hiding the cover, 277, 283
 - new job functionality, 285
 - private views, 282
 - programming solution, 276
 - resource image, 277
 - style features, 286
 - view applet, 280, 282
 - view HTML, 280, 282
 - view navigator, 279
 - workflow navigator, 285
- Web workflow
 - adjusting setup, 132
 - advanced programming, 275
 - advantages, 137
 - disadvantages, 138
 - enabling agents, 130
 - enabling workflow form, 130
 - process cache, 132, 133
 - standard Web interface, 134
- Workflow
 - components, 2
 - development, 10, 31

- events, 158
- life cycle, 6
- in Lotus Domino/Notes, 7
- reasons to use, 7
- view, 229
- where to use, 10

Workgroup documents

- alternative way to add
 - members, 70
- creating, 68

X

XML, 215

IBM Redbooks review

Your feedback is valued by the Redbook authors. In particular we are interested in situations where a Redbook “made the difference” in a task or problem you encountered. Using one of the following methods, **please review the Redbook, addressing value, subject matter, structure, depth and quality as appropriate.**

- Use the online **Contact us** review redbook form found at <http://www.redbooks.ibm.com/>
- Fax this form to: USA International Access Code +1 914 432 8264
- Send your comments in an Internet note to redbook@us.ibm.com

Document Number	SG24-5963-00
Redbook Title	Using Domino Workflow
Review	
What other subjects would you like to see IBM Redbooks address?	
Please rate your overall satisfaction?	<input type="radio"/> Very Good <input type="radio"/> Good <input type="radio"/> Average <input type="radio"/> Poor
Please identify yourself as belonging to one of the following groups:	<input type="radio"/> Customer <input type="radio"/> Business Partner <input type="radio"/> Solution Developer <input type="radio"/> IBM, Lotus or Tivoli Employee <input type="radio"/> None of the above
Your email address: The data you provide here may be used to provide you with information from IBM or our business partners about our products, services or activities.	<input type="checkbox"/> Please do not use the information collected here for future marketing or promotional contacts or other communications beyond the scope of this transaction.
Questions about IBM's privacy policy?	The following link explains how we protect your personal information. http://www.ibm.com/privacy/yourprivacy/

Printed in the U.S.A.

SG24-5963-00

Part No. CT6GNML

