

A Guide to Express

iPlanet™ Unified Development Server

Version 5.0

August 2001

Copyright (c) 2001 Sun Microsystems, Inc., 901 San Antonio Road, Palo Alto, California 94303, U.S.A. All rights reserved.

Sun Microsystems, Inc. has intellectual property rights relating to technology embodied in the product that is described in this document. In particular, and without limitation, these intellectual property rights may include one or more of the U.S. patents listed at <http://www.sun.com/patents> and one or more additional patents or pending patent applications in the U.S. and in other countries.

This document and the product to which it pertains are distributed under licenses restricting their use, copying, distribution, and decompilation. No part of the product or of this document may be reproduced in any form by any means without prior written authorization of Sun and its licensors, if any.

Third-party software, including font technology, is copyrighted and licensed from Sun suppliers.

Sun, Sun Microsystems, the Sun logo, Forte, iPlanet, Unified Development Server, and the iPlanet logo are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries.

UNIX is a registered trademark in the U.S. and other countries, exclusively licensed through X/Open Company, Ltd.

Federal Acquisitions: Commercial Software - Government Users Subject to Standard License Terms and Conditions.

DOCUMENTATION IS PROVIDED "AS IS" AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID.

Copyright (c) 2001 Sun Microsystems, Inc., 901 San Antonio Road, Palo Alto, California 94303, Etats-Unis. Tous droits réservés.

Sun Microsystems, Inc. a les droits de propriété intellectuels relatants à la technologie incorporée dans le produit qui est décrit dans ce document. En particulier, et sans la limitation, ces droits de propriété intellectuels peuvent inclure un ou plus des brevets américains énumérés à <http://www.sun.com/patents> et un ou les brevets plus supplémentaires ou les applications de brevet en attente dans les Etats - Unis et dans les autres pays.

Ce produit ou document est protégé par un copyright et distribué avec des licences qui en restreignent l'utilisation, la copie, la distribution, et la décompilation. Aucune partie de ce produit ou document ne peut être reproduite sous aucune forme, par quelque moyen que ce soit, sans l'autorisation préalable et écrite de Sun et de ses bailleurs de licence, s'il y en a.

Le logiciel détenu par des tiers, et qui comprend la technologie relative aux polices de caractères, est protégé par un copyright et licencié par des fournisseurs de Sun.

Sun, Sun Microsystems, le logo Sun, Forte, iPlanet, Unified Development Server, et le logo iPlanet sont des marques de fabrique ou des marques déposées de Sun Microsystems, Inc. aux Etats-Unis et dans d'autres pays.

UNIX est une marque déposée aux Etats-Unis et dans d'autres pays et licenciée exclusivement par X/Open Company, Ltd.

LA DOCUMENTATION EST FOURNIE "EN L'ÉTAT" ET TOUTES AUTRES CONDITIONS, DECLARATIONS ET GARANTIES EXPRESSES OU TACITES SONT FORMELLEMENT EXCLUES, DANS LA MESURE AUTORISEE PAR LA LOI APPLICABLE, Y COMPRIS NOTAMMENT TOUTE GARANTIE IMPLICITE RELATIVE A LA QUALITE MARCHANDE, A L'APTITUDE A UNE UTILISATION PARTICULIERE OU A L'ABSENCE DE CONTREFAÇON.

Contents

List of Figures	11
List of Procedures	15
Preface	19
Product Name Change	19
Audience for This Guide	20
Organization of This Guide	20
Text Conventions	21
Other Documentation Resources	21
iPlanet UDS Documentation	22
Express Documentation	22
WebEnterprise and WebEnterprise Designer Documentation	23
Online Help	23
iPlanet UDS Example Programs	23
Viewing and Searching PDF Files	24
Chapter 1 Using Express	27
Before Using Express	27
Entering Express	28
Using the iPlanet UDS Workshops	31
Chapter 2 Express Concepts	33
Overview	33
Business Models and Business Classes	35
The Object Modeling Technique	35
Association Symbols	37
Application Models and Business Class Windows	38
Code Generation	38
Partitioning and Deployment	39
Repository Use	40

About Express Workshops	41
Business Model Workshop	41
Application Model Workshop	42
Chapter 3 Express Tutorial	43
Overview	43
Setting Up Your Example Environment	44
Sample Table Descriptions	44
Building a Business Model	45
Importing Database Schema	45
About Importing Business Classes	48
Creating Associations	49
Aggregate Associations	50
Completing the Model	51
Generating the Server Classes	52
Creating an Application Model	52
Business Class Windows	55
Generating Window Classes	55
Running the CustomerOrder Window	55
Retrieving Records	57
Viewing Generated TOOL Classes	59
Creating a Simple Window Based on Two Tables	61
Adding Fields to a Window	62
Modifying Field Order	64
Using Multiple Windows	66
Creating a Lookup Link	67
Using the CustomerOrder Window	71
Enhancing the Application	73
Creating a Nested Link	73
Using the CustomerOrder Window	78
Customizing	81
Changing the Format of a Data Entry Field	81
Changing the Default Mode of a Window	86
Chapter 4 Using the Repository Workshop	89
Checking Out and Branching Models	89
Checking Out a Plan	90
Branching a Plan	91
How to Undo Changes	92
Compiling Plans	93
Repository Workshop Preferences	93

Chapter 5 Using the Business Model Workshop	95
About Business Models	95
Using the Business Model Workshop	96
Creating a New Business Model	96
Accessing Existing Business Models	97
Adding Supplier Plans	98
Supplier Plans and Generated Projects	98
The Business Model Workshop Window	99
Using the Palette	100
Using the Toolbar	101
Modeling Tools	101
Editing Tools	102
Viewing Tools	104
Alignment Tools	106
Viewing Workshop Components	107
Controlling Fonts	108
Access to Other Workshops	109
Leaving the Business Model Workshop	109
Business Classes	109
Creating a Business Class	109
Business Class/Service List	110
Setting Class Properties	111
Attributes	112
Primary Keys	112
Attribute Types	113
Field Label Text and Width	113
Custom Attributes	115
Creating an Attribute	115
Assigning Custom Domains	116
Editing Existing Attributes	117
Associations	117
Associations and Generated Classes	118
Direction	118
Multiplicity	119
To-One Associations	119
To-Many Associations	120
Optional Associations	121
Aggregation	121
Default Direction and Multiplicity	122
Creating an Association	123
Setting Association Properties	124
Creating a Generalization	125

Express Services	125
General Properties	127
Database Properties	129
Connection Properties	129
Creating New Express Services	129
Optional Environment Variables	132
Assigning Classes to a Service	133
Rules for Assigning Classes to Services	134
Editing Service Properties	134
Deleting Services	134
Importing and Exporting Database Schema	135
Importing Database Schema	135
Importing Schema from Multiple Databases	136
Exporting Database Schema	136
Relationship Between Business Classes and Exported Database Schema	137
Exporting Schema from Multiple Services	139
Server Code Generation	140
Validating the Model Prior to Generation	140
Potential Error Messages for Business Model Validation	141
Setting a Custom Prefix for Generated Classes	142
Generating Server Code	143
Forcing Complete Code Generation	145
Setting Workshop Preferences	145
Workshop Size and Position	146
Capturing Current Workshop Settings	146
Setting Dimensions of the Drawing Area	147
Specifying Workshop Position	147
Viewing Preferences	148
Drawing Preferences	149
Compiling Preferences	149
Font Preference	149
Command Summary	150
File Menu	150
Edit Menu	151
Component Menu	152
View Menu	153
Arrange Menu	153

Chapter 6 Using the Application Model Workshop	155
About the Application Model Workshop	156
Application Model Components	156
Window Layout	156
Window Behavior	157
Window Links	157
Start Window	158
Using the Application Model Workshop	158
Creating a New Application Model	158
Accessing Existing Application Models	159
Adding Supplier Plans	160
Supplier Plans and Generated Projects	161
The Application Model Workshop Window	162
Using the Palette	163
Using the Toolbar	164
Modeling Tools	164
Editing Tools	165
Viewing Tools	167
Alignment Tools	169
Viewing Workshop Components	170
Controlling Fonts	170
Access to Other Workshops	171
Leaving the Application Model Workshop	171
Business Class Windows	172
Creating a Business Class Window	172
Specifying Window Layout	174
Window Style	174
Field Placement	175
Choosing a Command Interface	176
Choosing a Command Set	177
Command Sets and Outline Windows	178
Creating Flexible Windows	179
Setting Window Properties	180
Editing Window Properties	180
Specifying Window Fields in the Attribute List	180
Changing Field Order	182
Specifying Window Fields from Two Business Classes	183
Changing Field Width	184
Specifying a Start Window	184

Links	185
Creating a Link	186
Link Types	186
Determining Data to Display in Called Window	187
Associations and Links	189
Fetching Detail Data	190
Returning Data From Lookup Links	193
Called Window Properties	193
Setting Link Properties	194
About Nested Links	194
Making Nested Windows Updateable	195
Creating Multiple Links to the Same Window	197
Example Flexible Window	197
Including an Existing Application Model in the Current Application Model	200
Supplier Plans	202
Using Callouts	202
Callouts to User-Defined Methods	202
Callouts to iPlanet UDS Projects	203
Callout to an Applet	205
Linking a Callout	206
Callouts to User-Defined Methods and Generated TOOL Code	207
Setting Default Behavior for Generated Windows	208
Client Code Generation	209
Validating the Model Prior to Generation	209
Potential Errors Found During Model Validation	209
Setting a Custom Prefix for Generated Classes	211
Forcing Complete Generation	212
Generating Client Code	213
Testing an Application	214
Setting Workshop Preferences	215
Workshop Size and Position	216
Capturing Current Workshop Settings	216
Explicitly Setting Window Dimensions	216
Specifying Workshop Position	217
Viewing Preferences	218
Drawing Preferences	218
Business Class Window Dialog Preferences	218
Fonts Preferences	219
Command Summary	220
File Menu	220
Edit Menu	221
Component Menu	221
View Menu	222
Arrange Menu	222

Chapter 7 Partitioning and Deployment	223
Partitioning Express Applications	223
About Express Projects and Service Objects	224
TheBroker Service Object	225
Service and DBService Service Objects	225
Service Service Objects	226
DBService Service Objects	227
Relationship between Service and DBService Service Objects	227
Default Partitioning for Express Applications	228
Testing the Application In a Distributed Environment	230
Deploying the Application	231
Modifying the Configuration	232
Modifying Service Object Definitions	232
Moving Service Objects	233
Replicating Partitions	234
Sharing a Business Server Between Applications	234
Chapter 8 Using Generated Windows	237
Using Windows	237
About Result Sets	237
About Window Modes	237
Simple Window Types	239
Form Window	239
Array Window	239
Outline Window	239
Inserting Records	240
Deleting Records	241
Retrieving Data	241
Scrolling	244
Clearing the Result Set	244
Reverting to Last Saved	245
Saving Changes to the Database	245
Using Links	245
Command Links	246
Drilldown Links	246
Lookup Links	247
Nested Links	248
Index	249

List of Figures

Figure 1-1	Express Icon	28
Figure 2-1	Three-Tier Architecture	34
Figure 2-2	Business Model Elements	36
Figure 2-3	Application Model Elements	38
Figure 2-4	Generated Projects	39
Figure 2-5	Business Model	41
Figure 2-6	Application Model	42
Figure 3-1	Tutorial Business Model	51
Figure 3-2	CustomerOrder Window	56
Figure 3-3	Customer Attributes in CustomerOrder Attribute List	62
Figure 3-4	CustomerOrder Window with Lookup Link to the Customer Window	71
Figure 3-5	CustomerOrder Window with Nested ListItem Window	78
Figure 4-1	Repository Workshop Preferences Dialog	93
Figure 5-1	Business Model Workshop Window	99
Figure 5-2	Business Model Workshop Toolbar Icons	101
Figure 5-3	Zoom Dialog	105
Figure 5-4	Grid Dialog	105
Figure 5-5	Grid Lines	106
Figure 5-6	Example of Multiplicity—Many to One	120
Figure 5-7	Example of Multiplicity—One to Many	120
Figure 5-8	Example of Optional Association	121
Figure 5-9	Aggregation	122
Figure 5-10	Example of Multiplicity—Many to One	122
Figure 5-11	One-to-Many Aggregation	123
Figure 5-12	MyModel Default Service	126

Figure 5-13	Classes Managed by MyModel Service	126
Figure 5-14	All Classes Assigned to One Service	133
Figure 5-15	Assigning a Class to a New Service	133
Figure 5-16	Tutorial Schema	138
Figure 6-1	Application Model Workshop Window	163
Figure 6-2	Application Model Workshop Toolbar Icons	164
Figure 6-3	Zoom Dialog	167
Figure 6-4	Grid Dialog	168
Figure 6-5	Grid Lines	168
Figure 6-6	Form Window	174
Figure 6-7	Array Window	175
Figure 6-8	Outline Window	175
Figure 6-9	Direction Property Set to Across	176
Figure 6-10	Wrap Property Set to 2	176
Figure 6-11	Attribute List	181
Figure 6-12	Many to One Association Between CustomerOrder and Customer	183
Figure 6-13	CustomerOrder Attribute View List Contains Customer Attributes	184
Figure 6-14	CustomerOrder to Customer—Many-to-One Association	191
Figure 6-15	CustomerOrder to LineItem—One-to-Many Association	192
Figure 6-16	Form Window Nesting an Outline Window	195
Figure 6-17	Array Window Nesting an Array Window	195
Figure 6-18	Form Window Nesting an Outline Window With a Button Interface	196
Figure 6-19	CustomerOrder Window	198
Figure 6-20	Multiple Links to Customer Window	198
Figure 6-21	Link Properties Dialog for Lookup Link	199
Figure 6-22	Link Properties Dialog for Command Link	199
Figure 6-23	Lookup Customer Window	200
Figure 6-24	Edit Customer Window	200
Figure 6-25	Included Application in Application Model	201
Figure 7-1	Relationship between Service and DBService Service Objects	227
Figure 7-2	Default Configuration For TutorialApp	229

Figure 8-1	Edit Mode Commands	238
Figure 8-2	Search Mode Commands	238
Figure 8-3	Form WIndow	239
Figure 8-4	Array Window	239
Figure 8-5	Outline Window	240
Figure 8-6	Command Link	246
Figure 8-7	Drilldown Link	247
Figure 8-8	Lookup Link	247
Figure 8-9	Nested Link	248
Figure 8-10	Folder Link	248

List of Procedures

- To copy the documentation to a client or server 24
- To view and search the documentation 25
- To open the Business Model Workshop 45
- To import database schema 46
- To create an association between CustomerOrder and Customer 49
- To create an aggregate association between CustomerOrder and LineItem 50
- To generate server classes 52
- To open the Application Model Workshop 53
- To create a window class based on the CustomerOrder business class 53
- To generate window classes 55
- To run the CustomerOrder window 56
- To retrieve a specific set of records (query by example) 57
- To retrieve all records 58
- To insert a record 58
- To retrieve the record you just entered 59
- To view the CustomerOrderWindow in the Window Workshop 59
- To add Customer attributes to the CustomerOrder window 63
- To change field order 64
- To run the window 66
- To create a window based on the Customer business class 67
- To create a lookup link 68
- To apply the Search command set and the Buttons default interface 70
- To use the lookup link to insert a record for an existing customer 72
- To create a nested link 73

To enter a new order for an existing customer	78
To retrieve a customer order	80
To insert a new line item on an existing order	80
To apply a template to a field	82
To view the generated TutorialApp classes	85
To change the default mode of a window	86
To check out a business or application model	90
To branch a business or application model	91
To revert a model	92
To undelete a model	92
To create a new business model	96
To modify an existing model in the Business Model Workshop	97
To add a supplier plan	98
To cut, copy, or delete an element	103
To paste an element	103
To find a business class	104
To align business classes	107
To set fonts for the current business model	108
To create a new business class	110
To create multiple classes without reselecting the New Class tool	110
To set or modify class properties	111
To create an attribute	115
To create multiple attributes	116
To assign a custom domain to an attribute	116
To draw an association between two classes	123
To create multiple associations without reselecting the New Association tool	124
To set association properties	124
To create a generalization	125
To create a service	130
To set database properties	131
To specify a connection option	131
To delete an empty service	134
To import database schema	135
To import schema from multiple databases	136

To export database schema definitions	139
To validate the business model	140
To enter a custom generation prefix	142
To generate server code	144
To set Business Model Workshop Preferences	145
To set the font for the Business Model Workshop	149
To create a new application model	159
To modify an existing application model in the Application Model Workshop	160
To specify a supplier plan for an application model	161
To cut, copy, or delete an element	166
To paste an element	166
To find a window class	166
To use an alignment tool	170
To set fonts for the current application model	171
To create a window class by dragging a Business Model class into the Application Model Workshop	172
To create a window class using the Application Model Workshop palette	173
To create multiple window classes without reselecting the New Class Window tool	173
To set window properties	180
To set attribute view properties	182
To change field order	182
To specify a start window	185
To create a link	186
To set link properties	194
To include an existing application model	200
To create a callout to a custom method	202
To create a callout to a project	204
To create a callout to an applet	205
To create a link to a callout	206
To set the callout link's properties	207
To set generated preferences	208
To validate a model	209
To enter a custom generation prefix	212

To generate client code	214
To set Application Model Workshop Preferences	215
To set the font for the Application Model Workshop	219
To test a client application from the Repository Workshop	230
To test a client application from the Partition Workshop	230
To deploy your Express application	231
To change the service object definition	232
To move a service object	233
To assign a logical partition	234
To include a shared service in your application	235
To make a reference partition	236

Preface

A Guide to Express provides complete information on the iPlanet UDS model-based application development tool, Express. This manual describes how to use the Express Application Model Workshop and Business Model Workshop.

This preface contains the following sections:

- “Product Name Change” on page 19
- “Audience for This Guide” on page 20
- “Organization of This Guide” on page 20
- “Text Conventions” on page 21
- “Other Documentation Resources” on page 21
- “iPlanet UDS Example Programs” on page 23
- “Viewing and Searching PDF Files” on page 24

Product Name Change

Forte 4GL has been renamed the iPlanet Unified Development Server. You will see full references to this name, as well as the abbreviations iPlanet UDS and UDS.

Audience for This Guide

This manual is intended for application developers. We assume that you:

- have programming experience
- are familiar with SQL and your particular database management system
- understand the basic concepts of object-oriented programming as described in *A Guide to the iPlanet UDS Workshops*
- have used the iPlanet UDS Repository Workshop

Organization of This Guide

The following table briefly describes the contents of each chapter:

Chapter	Description
Chapter 1, "Using Express"	Provides instructions for starting Express.
Chapter 2, "Express Concepts"	Defines the basic concepts of Express.
Chapter 3, "Express Tutorial"	A tutorial that walks you step by step through the process of creating an Express application.
Chapter 4, "Using the Repository Workshop"	Provides detailed information about using the iPlanet UDS Repository Workshop when you are working with Express.
Chapter 5, "Using the Business Model Workshop"	Provides detailed information about using the Business Model Workshop.
Chapter 6, "Using the Application Model Workshop"	Provides detailed information about using the Application Model Workshop.
Chapter 7, "Partitioning and Deployment"	Describes how to partition, test run, and deploy an Express application.
Chapter 8, "Using Generated Windows"	Provides information about how to use the windows that are generated by Express.

Text Conventions

This section provides information about the conventions used in this document.

Format	Description
<i>italics</i>	Italicized text is used to designate a document title, for emphasis, or for a word or phrase being introduced.
monospace	Monospace text represents example code, commands that you enter on the command line, directory, file, or path names, error message text, class names, method names (including all elements in the signature), package names, reserved words, and URLs.
ALL CAPS	Text in all capitals represents environment variables (FORTE_ROOT) or acronyms (UDS, JSP, iMQ). Uppercase text can also represent a constant. Type uppercase text exactly as shown.
Key+Key	Simultaneous keystrokes are joined with a plus sign: Ctrl+A means press both keys simultaneously.
Key-Key	Consecutive keystrokes are joined with a hyphen: Esc-S means press the Esc key, release it, then press the S key.

Other Documentation Resources

In addition to this guide, there are additional documentation resources, which are listed in the following sections. The documentation for all iPlanet UDS products (including Express, WebEnterprise, and WebEnterprise Designer) can be found on the iPlanet UDS Documentation CD. Be sure to read “[Viewing and Searching PDF Files](#)” on page 24 to learn how to view and search the documentation on the iPlanet UDS Documentation CD.

iPlanet UDS documentation can also be found online at <http://docs.iplanet.com/docs/manuals/uds.html>.

The titles of the iPlanet UDS documentation are listed in the following sections.

iPlanet UDS Documentation

- *A Guide to the iPlanet UDS Workshops*
- *Accessing Databases*
- *Building International Applications*
- *Esript and System Agent Reference Guide*
- *Fscript Reference Guide*
- *Getting Started With iPlanet UDS*
- *Integrating with External Systems*
- *iPlanet UDS Java Interoperability Guide*
- *iPlanet UDS Programming Guide*
- *iPlanet UDS System Installation Guide*
- *iPlanet UDS System Management Guide*
- *Programming with System Agents*
- *TOOL Reference Guide*
- *Using iPlanet UDS for OS/390*

Express Documentation

- *A Guide to Express*
- *Customizing Express Applications*
- *Express Installation Guide*

WebEnterprise and WebEnterprise Designer Documentation

- *A Guide to WebEnterprise*
- *Customizing WebEnterprise Designer Applications*
- *Getting Started with WebEnterprise Designer*
- *WebEnterprise Installation Guide*

Online Help

When you are using an iPlanet UDS development application, press the F1 key or use the Help menu to display online help. The help files are also available at the following location in your iPlanet UDS distribution:

```
FORTE_ROOT/userapp/forte/cln/*.hlp.
```

When you are using a script utility, such as Fscript or Escript, type help from the script shell for a description of all commands, or help *<command>* for help on a specific command.

iPlanet UDS Example Programs

A set of example programs is shipped with the iPlanet UDS product. The examples are located in subdirectories under `$FORTE_ROOT/install/examples`. The files containing the examples have a `.pex` suffix. You can search for TOOL commands or anything of special interest with operating system commands. The `.pex` files are text files, so it is safe to edit them, though you should only change private copies of the files.

Viewing and Searching PDF Files

You can view and search iPlanet UDS documentation PDF files directly from the documentation CD-ROM, store them locally on your computer, or store them on a server for multiuser network access.

NOTE You need Acrobat Reader 4.0+ to view and print the files. Acrobat Reader with Search is recommended and is available as a free download from <http://www.adobe.com>. If you do not use Acrobat Reader with Search, you can only view and print files; you cannot search across the collection of files.

➤ **To copy the documentation to a client or server**

1. Copy the `doc` directory and its contents from the CD-ROM to the client or server hard disk.

You can specify any convenient location for the `doc` directory; the location is not dependent on the iPlanet UDS distribution.

2. Set up a directory structure that keeps the `udsdoc.pdf` and the `uds` directory in the same relative location.

The directory structure must be preserved to use the Acrobat search feature.

NOTE To uninstall the documentation, delete the `doc` directory.

► **To view and search the documentation**

1. Open the file `udsdoc.pdf`, located in the `doc` directory.
2. Click the Search button at the bottom of the page or select `Edit > Search > Query`.
3. Enter the word or text string you are looking for in the Find Results Containing Text field of the Adobe Acrobat Search dialog box, and click Search.

A Search Results window displays the documents that contain the desired text. If more than one document from the collection contains the desired text, they are ranked for relevancy.

NOTE For details on how to expand or limit a search query using wild-card characters and operators, see the Adobe Acrobat Help.

4. Click the document title with the highest relevance (usually the first one in the list or with a solid-filled icon) to display the document.

All occurrences of the word or phrase on a page are highlighted.

5. Click the buttons on the Acrobat Reader toolbar or use shortcut keys to navigate through the search results, as shown in the following table:

Toolbar Button	Keyboard Command
Next Highlight	Ctrl+]]
Previous Highlight	Ctrl+[[
Next Document	Ctrl+Shift+]]

To return to the `udsdoc.pdf` file, click the Homepage bookmark at the top of the bookmarks list.

6. To revisit the query results, click the Results button at the bottom of the `udsdoc.pdf` home page or select `Edit > Search > Results`.

Using Express

This chapter describes in general how to start Express. Topics included in this chapter are:

- entering Express
- Express projects
- command line syntax

Before Using Express

Before you can use Express, you must have a full installation of iPlanet UDS 3.0 or higher.

To use Express, your workspace must include the following Express projects:

- AppModel
- ExpressDomains
- ExpressServices
- ExpressWindows
- ExpressDBBrowser

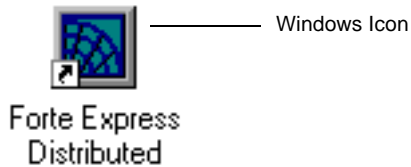
These projects are included automatically in FirstWorkspace and any workspaces you create subsequently.

Entering Express

Depending on your installation, you can enter Express either by selecting and opening a window system icon or by using a command line, as described below.

Figure 1-1 illustrates a window system icon for entering Express:

Figure 1-1 Express Icon



On systems that recognize a command line, you can use the following syntax to start Express. For Windows, you create a command line document and icon from the following command line.

Portable

```
express [-fs] [-fr repository] [-fw workspace] [-fnd node_name]
          [-fmn model_node_name] [-fm memory_flags] [-fl logger_flags]
[-fcons]
```

OpenVMS

```
VFORTE EXPRESS
  [/STANDALONE]
  [/REPOSITORY=repository_name]
  [/WORKSPACE=workspace_name]
  [/NODE=node_name]
  [/MODEL_NODE=model_node_name]
  [/MEMORY=memory_flags]
  [/LOGGER=logger_flags]
  [/FCONS]
```

The following table explains each of the command line flags:

This Flag	Specifies
-fs /STANDALONE	Run the development session in standalone mode. In standalone mode, the node is not connected to the distributed development environment, and you cannot partition projects in the Partition Workshop or use a central repository. The default mode is distributed, which connects you to the name server and environment manager (see the <i>iPlanet UDS System Management Guide</i> for information about the distributed development environment).
-fr <i>repository</i> /REPOSITORY= <i>repository_name</i>	The repository to be used for the development session. See below for information on specifying a repository.
-fw <i>workspace</i> /WORKSPACE= <i>workspace_name</i>	The workspace for the development session. See below for information on specifying a workspace.
-fnd <i>node_name</i> /NODE= <i>node_name</i>	Specifies the node name to use for this session. If you do not specify the node name in the express command, the default node name depends on the operating system. On Windows, the default node name is set by the FORTE_NODENAME environment variable. On all other platforms, the actual node name is used.
-fmn <i>model_node_name</i> /MODEL_NODE= <i>model_node_name</i>	Specifies the model node name to use for this session. If you do not specify a model node name in the express command, iPlanet UDS uses the value of the FORTE_MODELNODE environment variable. If the environment variable is not set, the node is not treated as a model node.
-fm <i>memory_flags</i> /MEMORY= <i>memory_flags</i>	Specifies the space to use for the memory manager. See Appendix B in <i>A Guide to the iPlanet UDS Workshops</i> for details.

This Flag	Specifies
-fl <i>logger_flags</i> /LOGGER= <i>logger_flags</i>	Specifies the logger flags to use for the session. See Appendix B in <i>A Guide to the iPlanet UDS Workshops</i> for details. If you do not set the logger flags in the <code>express</code> command, iPlanet UDS uses the value of the <code>FORTE_LOGGER_SETUP</code> environment variable. Note that you can change the logger settings from the Repository Workshop.
-fcons /FCONS	Displays the trace window. By default, the trace window is iconized on Windows. Use this flag to display the trace window on startup.

The following are examples of the `express` command:

```
express -fs
express -fr ct:$FORTE_ROOT/repos/examples
express -fl "%stdout(trc:os:1:1 trc:err)" -fm "(n:4000,x:8000)"
```

Selecting a repository To specify the repository in the `express` command:

- For a central repository, specify a repository service name. See your system manager for information about your central repository.
- For a private C-tree repository, specify the repository name using the following format:
ct: *private_repository_name*.
- For a shadow repository, specify the repository name using the following format:
ct: *shadow_repository_name*.

See *A Guide to the iPlanet UDS Workshops* for information about the different kinds of repositories.

Default repository If you do not specify a repository in the express command, the default repository depends on the operating system. For all platforms, iPlanet UDS uses the setting of the FORTE_REPOSNAME environment variable, and if the environment variable is not set, iPlanet UDS uses the distributed repository called “CentralRepository.” For Windows, iPlanet UDS uses the last repository you opened in the Repository Workshop.

Selecting a workspace If you do not specify a workspace in the express command, the default workspace depends on the operating system. For all platforms, iPlanet UDS uses the setting of the FORTE_WORKSPACE environment variable, and if the environment variable is not set, iPlanet UDS opens the Repository Workshop without a workspace. For Windows, iPlanet UDS uses the last workspace you opened in the Repository Workshop. If the value of FORTE_WORKSPACE is not set, the default workspace is FirstWorkspace.

Using the iPlanet UDS Workshops

For general information about using the iPlanet UDS Workshops and user interface, see *A Guide to the iPlanet UDS Workshops*.

Entering Express

Express Concepts

Welcome to Express, the iPlanet UDS application generation tool. Using Express workshops—the Business Model Workshop and the Application Model Workshop—you develop graphical models of objects in your business system and the window flow of your application. Express automatically generates an application based on these models. Subsequently, you can use the standard iPlanet UDS workshops to further refine and customize your final application.

This chapter contains a high-level discussion of the basic concepts and facilities of Express. Topics covered include:

- overview of business models and the Business Model Workshop
- Object Modeling Technique concepts
- overview of application models and the Application Model Workshop
- the relationship between Express and the standard iPlanet UDS workshops

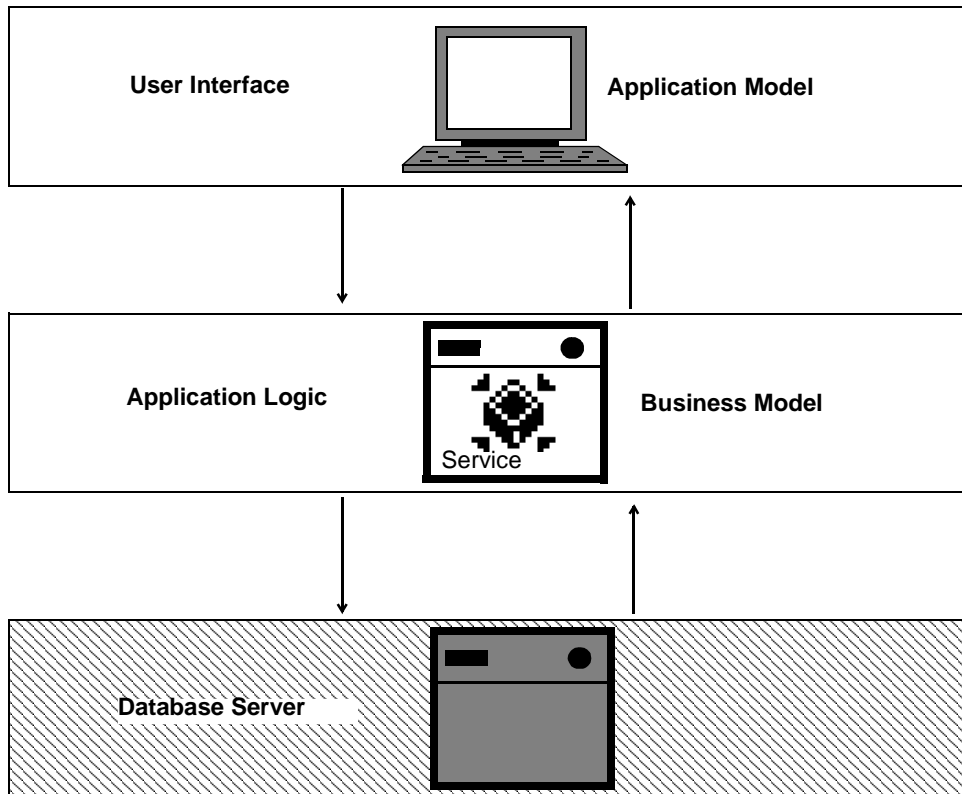
Overview

Express is model-based application development system that lets you create iPlanet UDS database query-update applications without writing TOOL code. To create an Express application, you use two workshops, the Business Model and Application Model Workshops, to draw graphical representations of your application.

To define the objects and services in your application, you create a *business model*. The business model is a graphical representation of the data in your application. To create the user interface for your application, you create an *application model*. The application model is a graphical representation of the windows in your user interface and the control flow of your application.

Three-tier architecture Express supports a three-tier architecture for your database applications. In a three-tier architecture, the user interface, application services, and database processing are divided into separate modules, as illustrated in the following figure:

Figure 2-1 Three-Tier Architecture



The primary advantage of this three-tier architecture is that you can partition your application so that the application services, as well as the client portions, run on different nodes than the database management system. With iPlanet UDS this also enables you to provide failover and load balancing across servers and machines.

Using Express, you can build both the first and second tiers of your application. You use the Application Model Workshop to create the first tier, the user interface, and you use the Business Model Workshop to create the second tier, the application services.

Partitioning When it comes time to deploy the application, you can treat it the same way you do any other iPlanet UDS application. You use the Partition Workshop to partition the application for each deployment environment, and create the distributions for each configuration you wish to deploy.

Customization Express is completely integrated with the rest of the iPlanet UDS development environment. If desired, you can customize the application you create with Express using other iPlanet UDS workshops. For example, you could use the Window Workshop to modify a window you created using Express. Changes you make in other iPlanet UDS workshops are reflected when you return to Express. You can also integrate non-Express projects into your Express applications, or use Express projects in standard iPlanet UDS applications. Refer to *Customizing Express Applications* for information about customization.

Business Models and Business Classes

The foundation of a Express application is a business model. A business model is a graphical representation of the data in the system you are building. Business models are comprised of business classes that correspond to the objects in your system, and the relationships between them. For example, if you are building an order-entry system, then some business classes in your model might be orders, customers, line items, parts, and so on. You may have a relationship between orders and customers, and orders and line items.

Business classes map directly to database tables—access to the database management system is automatic.

You construct business models in the Business Model Workshop. The Business Model Workshop is a graphical tool that supports a subset of the Object Modeling Technique (OMT) notation for modeling a business system (described below). Once you define the business classes, you will draw the relationships between them.

For more information about the Business Model Workshop, see [Chapter 5, “Using the Business Model Workshop.”](#)

The Object Modeling Technique

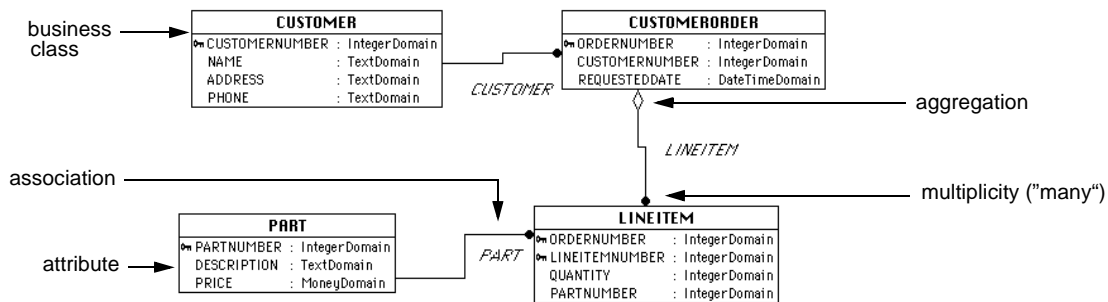
The Object Modeling Technique (OMT) is a graphical notation for representing object-oriented concepts. For more information about OMT, see *Object-Oriented Modeling and Design*, by James Rumbaugh and others. You use the tools in the Business Model Workshop to create OMT symbols that represent classes and their relationships.

The Business Model Workshop supports the following OMT concepts and graphical notations:

- class
- attribute
- association
- generalization
- aggregation
- inheritance
- multiplicity

Figure 2-2 illustrates the graphical elements of a business model.

Figure 2-2 Business Model Elements



Each element is described below.

Business Class A business class represents an object in your business system. Each class corresponds to a database table and to one or more TOOL classes. A business class is represented by a box containing a separate region for attributes.

Attribute An attribute is a data value held by the objects in a class. Attributes appear in the lower part of a class, below the class name. Every attribute corresponds to a column in a database table.

Association An association is the relationship, or connection, between two classes. Associations in the Business Model Workshop are uni-directional; but, you can draw multiple associations (in different directions) between two classes.





Multiplicity Multiplicity describes how many instances of one class may relate to a single instance of another class. For example, one customer can place many orders—this is a “one-to-many” relationship. Line terminators specify the type of relationship: a solid dot indicates zero or more—“many.” No dot means one—a line without multiplicity symbols means a one-to-one association.

Aggregation An aggregation is a special type of association that describes a relationship between the objects representing the components of a larger assembly and the object representing the entire assembly. Component objects cannot exist without the assembly object. A small diamond indicates the assembly end of the relationship. For example, the association between `CustomerOrder` and `LineItem` is an aggregate relationship—a line item cannot exist without an order. Therefore, `CustomerOrder` is the assembly class, and `LineItem` is the component class.

Generalization/Inheritance A generalization is a relationship between a class and one or more specific versions of the class. The more general class is the superclass, and the more specific class is its subclass. Inheritance is the mechanism through which subclasses share the features (attributes, methods) of their superclasses. A subclass “inherits” the attributes of its superclass. A generalization is indicated by a triangle on the line connecting the classes.

Association Symbols

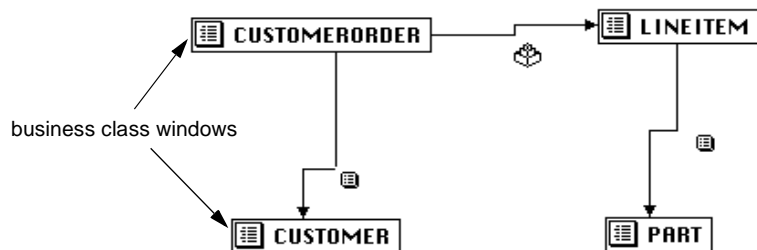
The following table displays the symbols that define types of associations:

Symbol	Description
	Direction of association. The class at the end with the arrow is referred to as the ‘to’ class. The class at the end without the arrow is the ‘from’ class.
	
●	Zero or more (many).
no symbol	One
	Aggregation
	Generalization

Application Models and Business Class Windows

Once you have described your business system in the Business Model Workshop, you can define the windows that will comprise the user interface of your application and specify the flow of control between windows. For example, [Figure 2-3](#) illustrates the relationship between windows in an application—the relationships are specified with links:

Figure 2-3 Application Model Elements

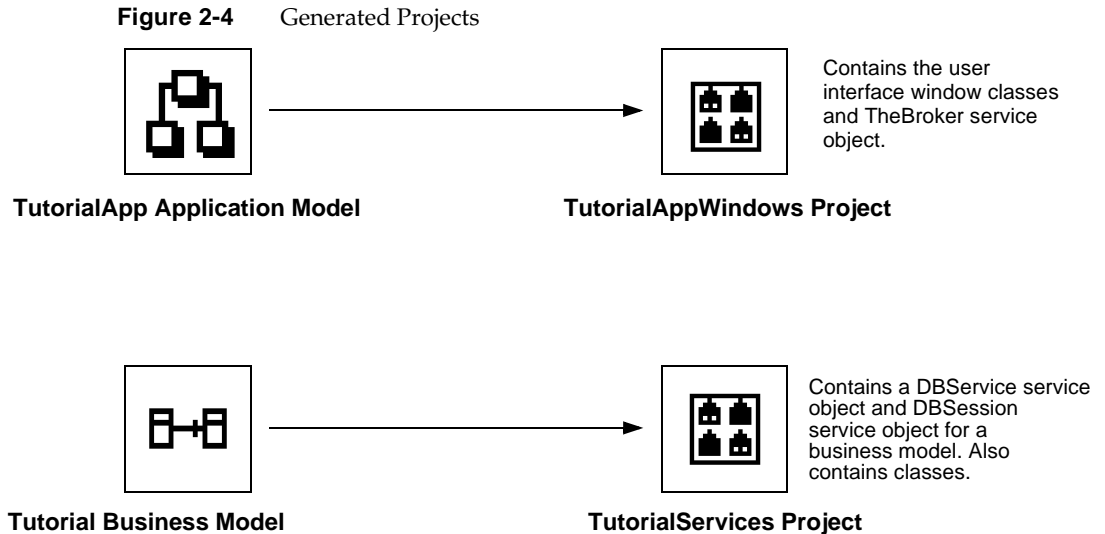


You create windows and specify control flow in the Application Model Workshop, where you build application models. An application model is comprised of business class windows and links, and, if desired, standard iPlanet UDS windows or even windows from external systems, such as Microsoft Excel. Every business class window is based on one business class from a corresponding business model. Windows can be nested, which allows you to manipulate data from multiple tables. In Business Class Window Properties dialogs, you can specify window layout, the type of database command interface, and window behavior.

Application models also reflect how windows communicate with each other, including how they are invoked and how data is passed between them.

Code Generation

Express automatically generates TOOL code based on the models you create in the Business Model and Application Model workshops. From the business model, Express generates the service objects and classes that handle the data in your application. From the application model, Express generates the iPlanet UDS window classes that provide your user interface. [Figure 2-4](#) shows the projects that are created from your business and application models:



The Windows project generated from the application model is the *main project* for the application. The Services project generated from the business models functions as a *supplier project* to the main project.

Once these iPlanet UDS projects are generated, you can modify them, if necessary, as you would any iPlanet UDS projects by using the standard iPlanet UDS workshops. If you change parts of the model, when you regenerate code, Express automatically regenerates the code to accommodate the changes, but maintains any custom changes you have made.

Partitioning and Deployment

Default partitioning Express automatically creates all the service objects needed for your application. When you test run the application in Express, iPlanet UDS partitions the application so that as much of it as possible will run on your client machine. Any service objects that the client node cannot support are assigned to a suitable node in your development environment, allowing you to test the application independently of its deployment environments.

Partition Workshop When the application is ready for deployment or, if you wish to test the application in a distributed environment, you use the Partition Workshop. At this point, you treat your Express application like any other iPlanet UDS application. You can partition the Windows project generated from your Express application model the same way you partition any other iPlanet UDS main project. Partitioning your Express application automatically constructs all three tiers of your database application.

You can then use the standard iPlanet UDS services to create the distribution, including compiling the distributed partitions for optimal performance.

Repository Use

The business models and application models that you create with Express are stored in your development repository the same way as iPlanet UDS projects you create with the Project Workshop—you must use the Repository Workshop to manage them. In the Repository Workshop, you can create new models, include existing models in your workspace or exclude them, and delete existing models from the repository. To get write access to individual models, you use Checkout and Branch commands.

In addition, all work that you perform in Express takes effect in the current workspace. You must be sure to save, update, and integrate your workspace as appropriate.

See [Chapter 4, “Using the Repository Workshop,”](#) for information about working with business models and application models with the Repository Workshop, and *A Guide to the iPlanet UDS Workshops* for general information about using iPlanet UDS workspaces.

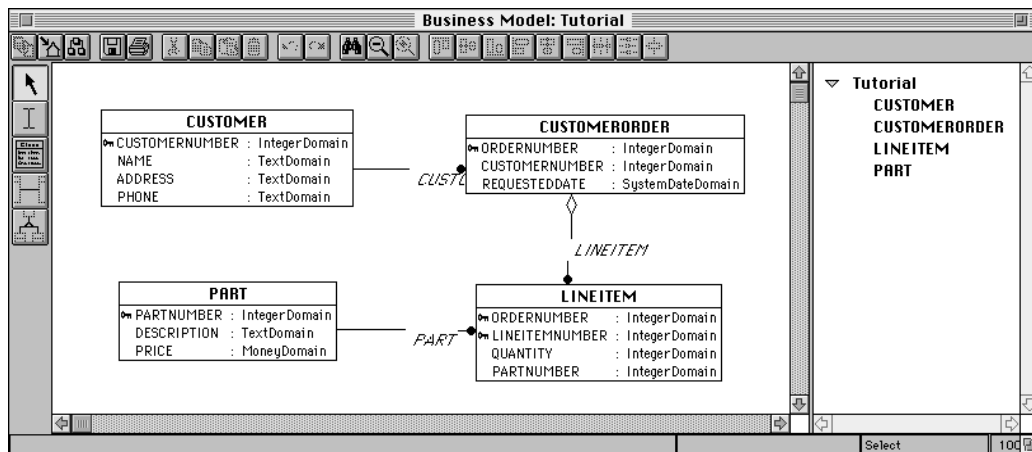
About Express Workshops

Express features two workshops: the Business Model Workshop and the Application Model Workshop. This section briefly introduces you to these workshops, preparing you to use the tutorial in the next chapter. The workshops are discussed in detail in [Chapter 5, “Using the Business Model Workshop,”](#) and [Chapter 6, “Using the Application Model Workshop.”](#)

Business Model Workshop

The Business Model Workshop provides the tools you need to draw and specify the components of a business model. A business model consists of business classes, attributes, associations, and generalizations, as shown in [Figure 2-5.](#)

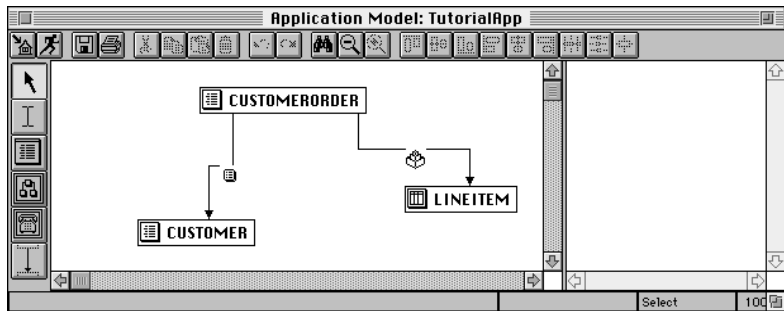
Figure 2-5 Business Model



Application Model Workshop

The Application Model Workshop provides the tools you need to build and connect the windows that will comprise the user interface of your application. An application model, shown in **Figure 2-6**, consists of windows, links—the mechanisms through which windows communicate—and properties that specify window layout and behavior.

Figure 2-6 Application Model



Express Tutorial

This chapter contains a tutorial that walks you through the creation an application with Express. The tutorial highlights a variety of concepts and features of Express, and provides cross references to sections in the manual where you can find more information.

Overview

The best way to learn about Express is to use it. The following tutorial steps you through the construction of an application, demonstrating many of the features of Express. Conceptual discussion is included where appropriate, as well as cross references to areas later in this manual where you can find more information.

In this tutorial, you will import a sample database schema to create the basis of your business model. Next you will create the relationships between the business classes and then build the application model.

The sample application is an order-entry system that allows a user to view the orders associated with each customer, enter new orders for existing customers, and enter new customers. Users can also view the line items that comprise an order, and enter new line items as they create new orders.

In this tutorial you will learn how to do the following:

- import database schema
- draw a business model
- generate server classes and service objects
- draw an application model
- generate window classes

- create a simple window based on one table
- create a simple window based on multiple tables
- create different data interfaces
- link windows
- perform a simple customization

The tutorial is organized into three parts.

- [“Building a Business Model” on page 45](#) focuses on building a business model.
- [“Creating an Application Model” on page 52](#) teaches you how to build and run a simple application.
- [“Enhancing the Application” on page 73](#) shows you how to add more complicated functionality to the application, as well as how to do some customization.

Setting Up Your Example Environment

Scripts for loading the sample database tables are located in `FORTE_ROOT/userapp/express/cl#/examples/install`. Use standard file redirection to load `maketut.ora` or `maketut.syb` into your database. Before loading the `maketut.syb` file, edit it to give it a database name of your choice. If you wish to drop the tables after you complete the tutorial, run `droptut.syb` or `droptut.ora`.

Sample Table Descriptions

Table Name	Columns
Customer	CustomerNumber (primary key) Name Address Phone
Customer Order	OrderNumber (primary key) CustomerNumber RequestedDate
LineItem	OrderNumber (primary key) LineItemNumber (primary key) Quantity PartNumber

Table Name	Columns
OrdUser	UserNumber (primary key) UserName Password
Part	PartNumber (primary key) Description Price

Building a Business Model

As mentioned earlier, this tutorial is based on the sample database included on your Express media. The first step in developing your application is to “drag and drop” the tables you wish to use into the Business Model Workshop. Each table you drag will become a business class in your business model. Any columns in the tables will become attributes in their respective business classes.

The business model is used to generate the server tier and, optionally, the database tier of your three-tier application.

Importing Database Schema

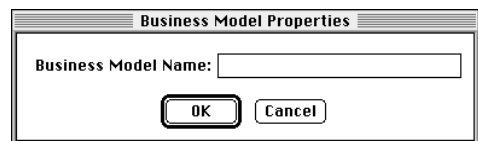
You import database tables from the Business Model Workshop.

► To open the Business Model Workshop

1. Start Express.
2. In the Repository Workshop, click the New Business Model icon.

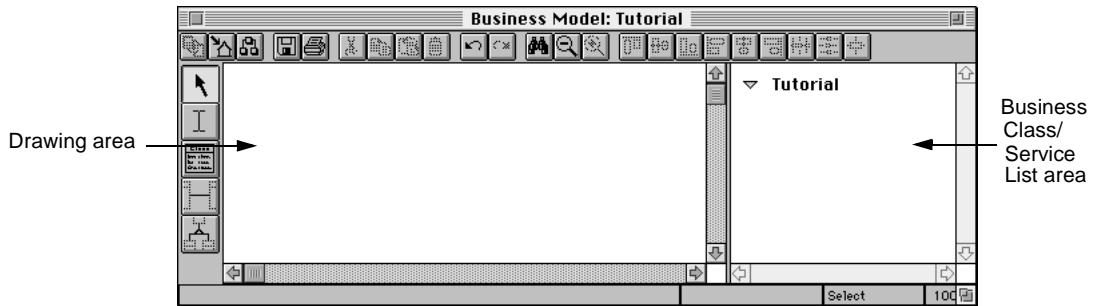


The Business Model Properties dialog appears, as shown in the figure below. Use this dialog to name the business model.



3. Type `Tutorial` in the Business Model Name field.

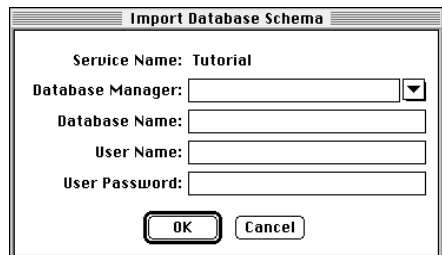
- Click OK and the Business Model Workshop appears, as shown in the figure below.



➤ **To import database schema**

- Choose the File > Import Database Schema command.

The Import Database Schema dialog appears, as shown below.

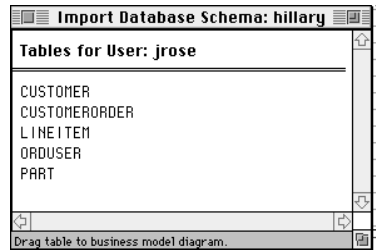


Each time you create a new business model, Express creates a default service. A service is a central point of access and control for a set of related business classes. The following properties give your service access to a set of database tables. See [Chapter 5, "Using the Business Model Workshop,"](#) for information about Express services.

- Type the name of your resource manager in the Database Manager field.
- Type the name of your database in the Database Name field.
- Type the user name for the account in which you set up the sample tables in the User Name field.
- Type the corresponding password for the user name in the Password field.

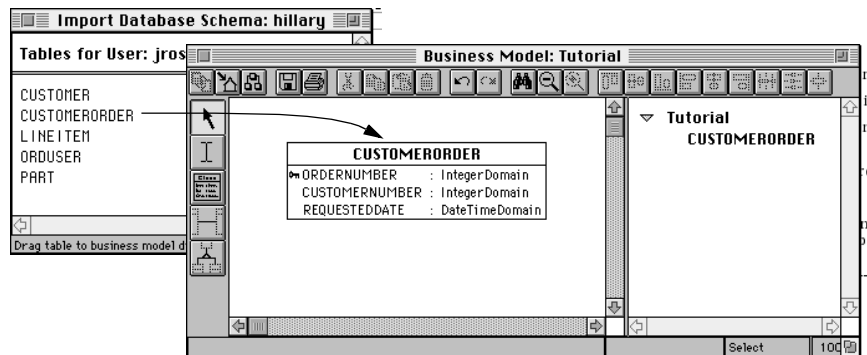
6. Click OK.

The Import Database Schema window appears, as shown below. This dialog contains a list of tables found in the database schema of the database specified in the Service Properties dialog.



7. Place the cursor over the table name “CustomerOrder,” hold down the mouse button, drag the table name into the Business Model Workshop, and then release the mouse button.

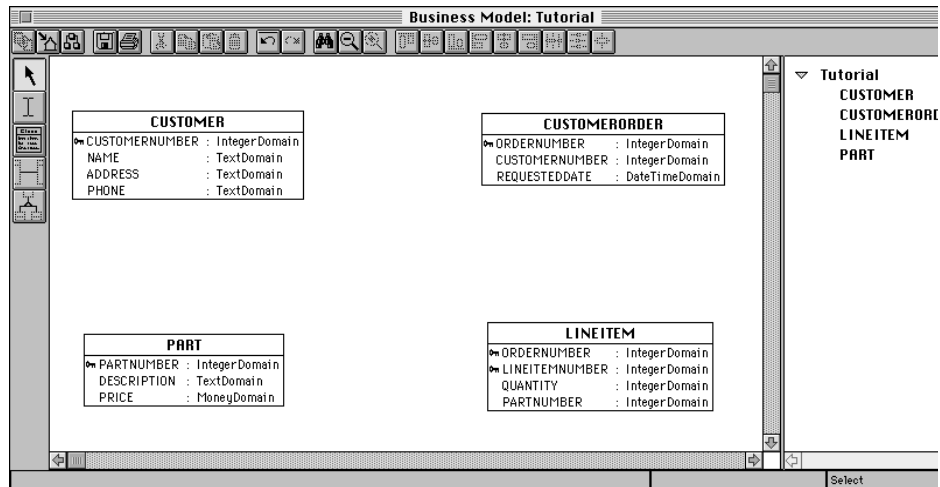
A business class named CustomerOrder appears—with all its attributes automatically defined—as shown below. Express creates an attribute for each column in the CustomerOrder table.



8. Repeat the process for the Customer, LineItem, and Part tables.

9. Close the Import Database Schema dialog.

Move the classes around the drawing area to resemble the screen below.



10. Click the Save All command.



About Importing Business Classes

As you can see, Express represents each table you imported as a business class. Each column in the table has a corresponding attribute in the business class.



If you import a table from a database that supports primary keys in the data definition language (DDL), then Express will maintain and represent the primary key in the business class (as is indicated by the key icon). There is a primary key in each business class—refer to [“Sample Table Descriptions” on page 44](#) to verify which attributes are primary keys. If your database does not support primary keys in the DDL, then you must create a primary key attribute for each business class in your business model. For information about creating primary keys, see [“Primary Keys” on page 112](#).

After you create your business classes, the next step is to create associations between them.

Creating Associations

You create relationships, or associations, between business classes using the New Association tool. An association describes how one business object relates to another. The way you draw associations affects how the underlying iPlanet UDS classes are generated.

In the generated iPlanet UDS classes, an association from one class to another class is represented by an attribute in the from-class that is used to hold a reference to an object of the to-class. The direction of the association specifies which class contains the reference to the other class. The multiplicity of an association determines if the attribute references a single object or an array of objects, or if many attributes can reference the same object. Generated classes are discussed in detail in *Customizing Express Applications*.

Express assigns default values for association properties—in this tutorial, the defaults do not need to be changed. For more information about association properties, see [Chapter 6, “Using the Application Model Workshop.”](#)

► To create an association between CustomerOrder and Customer

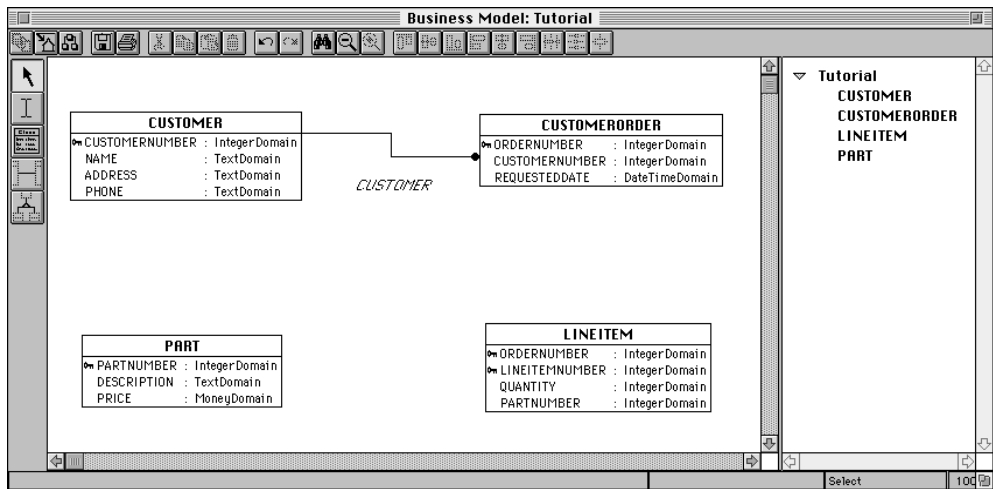
1. Click the New Association tool.



2. Click the Customer class and drag the association to the CustomerOrder class. Release the mouse button.

Note that when you are drawing associations, it does not matter which class you select first.

An association labeled CUSTOMER appears, as shown below.



Notice that the name of the association is CUSTOMER—the conventions for naming associations are described in *Customizing Express Applications*.

Aggregate Associations

In addition to an association's multiplicity and direction, associations have an Aggregation property. An aggregation is a type of relationship where one object is a component of another object. For example, an order is made of up line items—a line item is a component of an order. Another characteristic of an aggregate relationship is that the component object cannot exist without the other object—a line item cannot exist without a corresponding order. The aggregate class is the class that contains the component class, in this case, CustomerOrder. Aggregation is indicated by a diamond symbol at the aggregate class end of the association.

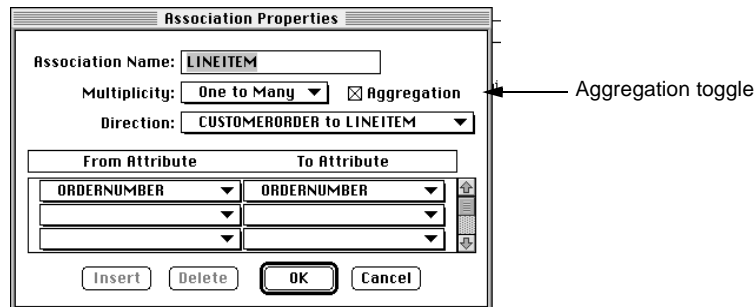
► To create an aggregate association between CustomerOrder and LineItem

1. Click the New Association tool.



2. Draw an association from CustomerOrder to LineItem.
3. Double-click on the association to display its Properties dialog.

4. Check the Aggregation toggle to create an aggregate association.



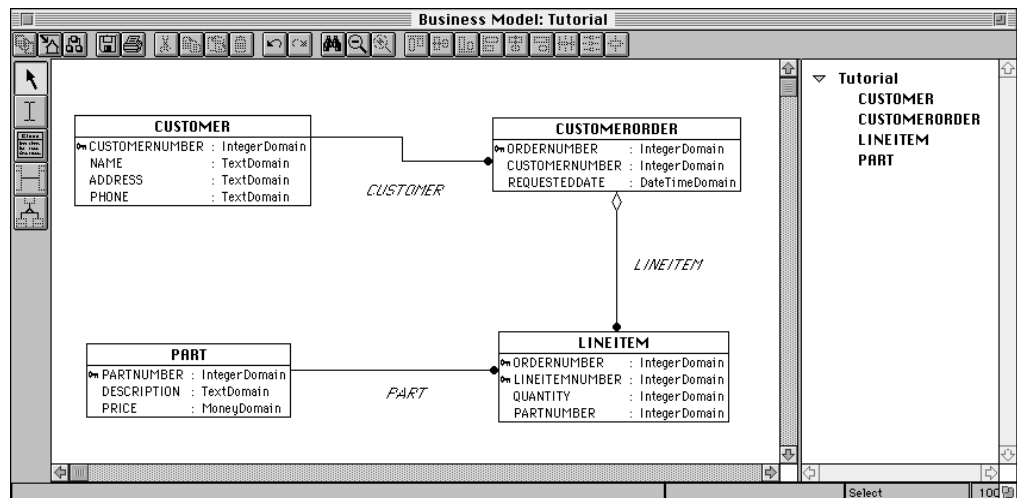
5. Click OK to create an aggregate association.

Completing the Model

Finally, to complete the business model, select the Association tool and draw a line from LineItem to Part.

Your business model should look like [Figure 3-1](#):

Figure 3-1 Tutorial Business Model



You now have a complete business model. Once you have established your business classes and their relationships, you can generate the underlying TOOL classes that will handle the data in your application.

Generating the Server Classes

The business classes you create represent the key business objects manipulated by the application. In Express, business objects are managed by services. These services provide the second tier of a three-tier application architecture. Express automatically creates the TOOL classes corresponding to the business classes as well as the service objects providing the services used to access the business classes. These services contain all the code required to access and update the database, as well as concurrency management capabilities required by a scalable application.

► **To generate server classes**

1. Select the File > Generate Server Code command, or click the Generate icon in the toolbar.



A status box appears, reporting the progress of the generation.

Refer to *Customizing Express Applications* for more information about the classes generated automatically by Express.

Your next step is to build your application model.

Creating an Application Model

This section shows you how to use the Application Model Workshop to create the client side of the application—the application model. The client consists of a set of windows used to view and edit application data, and the commands used to navigate from window to window. You can create multiple application models from one business model. You can also build an application model using classes in several business models.

Your first step is to create a window based on the CustomerOrder business class.

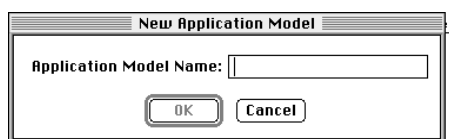
► **To open the Application Model Workshop**

1. Click the New Application Model icon in the Business Model Workshop toolbar.



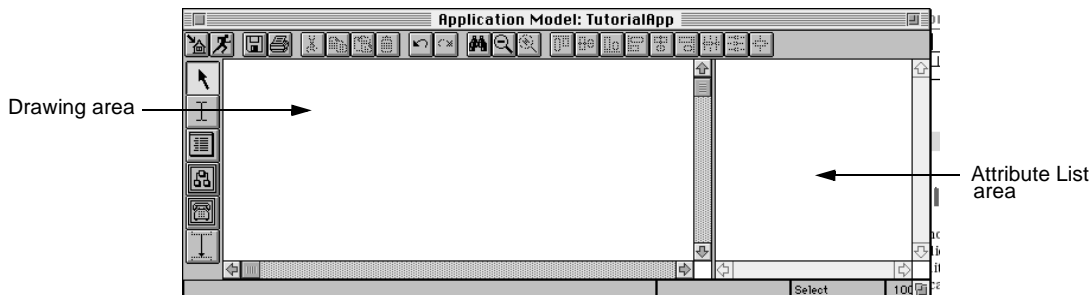
You can also click the New Application Model icon in the Repository Workshop, or choose the Plan > New Application Model... command.

The Application Model Properties dialog appears, as shown below.



2. Type the name TutorialApp, and then click OK.

The Application Model Workshop opens, as shown in the figure below.

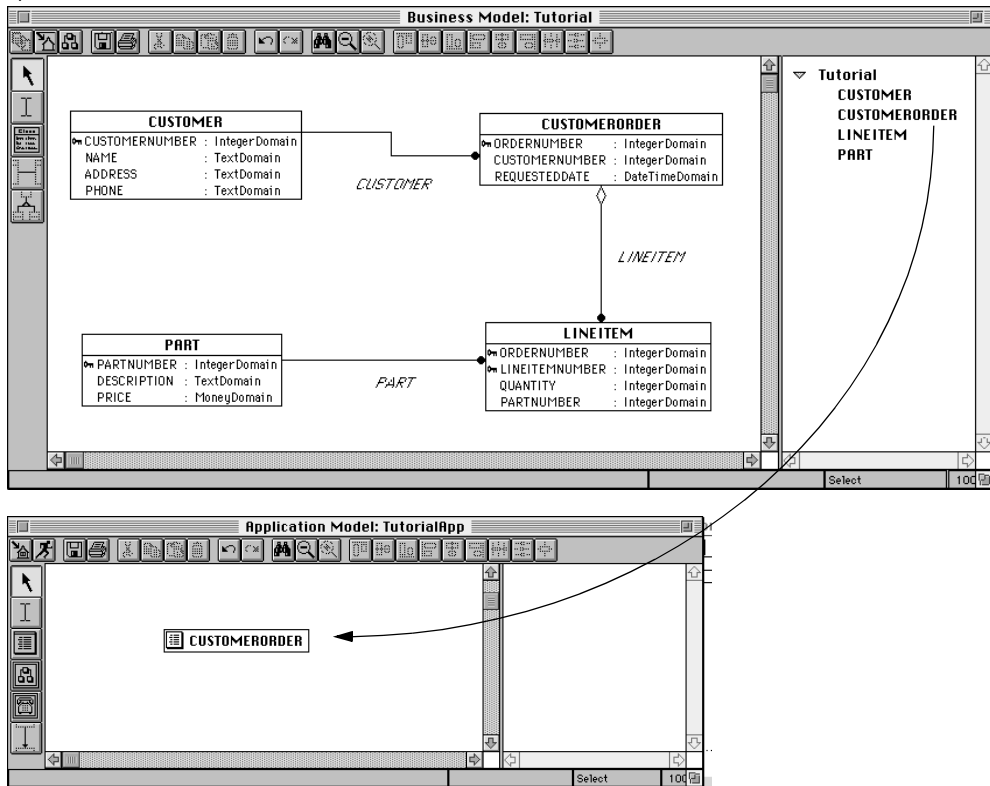


► **To create a window class based on the CustomerOrder business class**

1. In the Business Model Workshop, select CustomerOrder from the Business Class List.

The Business Class List is the right-hand panel that contains the list of business classes in the current model.

2. Drag the class from the list and drop it into the drawing area in the Application Model Workshop



An icon labeled CustomerOrder appears in the workshop, as shown in the figure above. CustomerOrder represents a window based on the CustomerOrder business class.

Business Class Windows

Express automatically creates a business class window, which represents an actual window in a generated application. A business class window is simply a window based on a business class in the business model. By default, all the attributes on the business class will appear as fields on the corresponding business class window at runtime. When you generate the business class window TOOL code, you will be able to view and modify the business class window in the Window Workshop, just like any other iPlanet UDS window. For more information about windows, see [Chapter 6, “Using the Application Model Workshop.”](#)

Generating Window Classes

At this point, the default settings for the CustomerOrder window are sufficient for the model. The next step is to generate the window classes based on the CustomerOrder window.

➤ **To generate window classes**

1. Click the Generate Client Code icon, or choose the File > Generate Client Code command.



A status box appears, reporting the progress of the generation.

Refer to *Customizing Express Applications* for information about the generated client classes.

Running the CustomerOrder Window

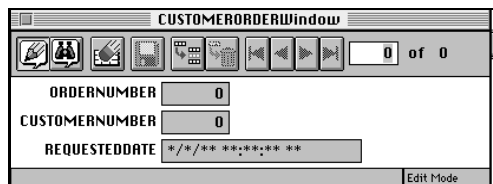
At this point, you have defined a business class, created its associated business class window, and have a fully specified window. Next, you can run the CustomerOrder window and examine the functionality Express provides by default. When you test run an application in Express, iPlanet UDS partitions the application so that as much of it as possible will run on your client machine. Any service objects that the client node cannot support are assigned to a suitable node in your development environment. This behavior allows you to test the application independently of its deployment environments. For information about testing an application in a distributed environment, see [Chapter 7, “Partitioning and Deployment.”](#)

► **To run the CustomerOrder window**

1. Click the Run Application icon.



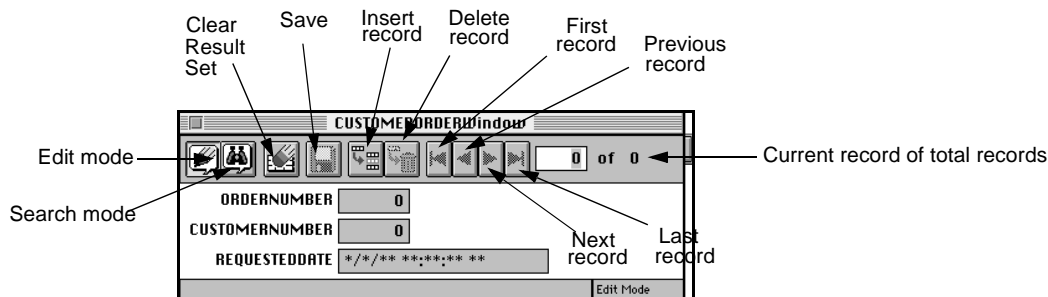
The CustomerOrder window appears, as shown below.



Generating classes As you might notice, both the Business Model Workshop and the Application Model Workshop have commands that generate all the classes associated with both models. You can generate the classes at any time. In addition, the Application Model Workshop always checks that the generated classes are up to date in both workshops before attempting to run an application. If you run an application before generating the classes (or if you make changes to either model and forget to regenerate), Express automatically generates the classes before running the application. Note that if you run the application from the Repository Workshop, you must explicitly generate the classes beforehand.

Figure 3-2 identifies all the commands that are generated by default:

Figure 3-2 CustomerOrder Window



The following sections will get you acquainted with the CustomerOrder window. For more information about using windows, see [Chapter 8, “Using Generated Windows.”](#)

Retrieving Records

Express applications provide a search mechanism that supports *query by example*. Query by example means that you can enter information in any field on the window, and the application will try to find a record that contains that information. For example, if you know you are looking for an order placed on a specific date, you can enter that date in the RequestedDate field, and your application will retrieve all records entered on that day.

Wildcard character Express supports the use of the wildcard character (%), which allows end users to use partial search criteria. For example, if the user types J% in the customer name field, Express returns all customers with names that begin with J. Conversely, using %J returns all customers whose names end with J. Any other wildcard mechanisms are database dependent.

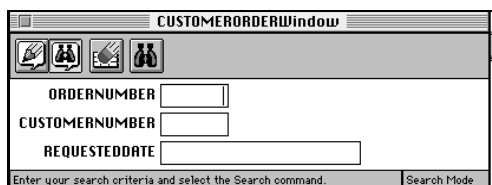
For a complete list of operators supported for each data type in Query by Example, see [“Retrieving Data” on page 241](#).

► To retrieve a specific set of records (query by example)

1. Click the Search Mode icon.



The text fields become editable, and the toolbar changes to display the Mode commands, the Clear Result Set command, and the Search command.



2. Type 1 in the Customer Number field and click the Search icon.

The application retrieves all records with Customer Number of 1. The first record will appear on the form; the Current Record of Total Records indicator displays how many records were retrieved.

► **To retrieve all records**

1. Click the Search mode icon.



2. Click the Search icon.



The application retrieves all records in the CustomerOrder table, and displays the first record.

Scrolling through records To scroll through the records one by one, click the Next Record icon. To scroll to the last record, click the Last Record icon; First Record returns to the first record.

Inserting and Deleting Records

You can insert a new record into the CustomerOrder table using the CustomerOrder window.

► **To insert a record**

1. Click the Insert Record icon.



2. Type 100 in the OrderNumber field.
3. Type 33 in the CustomerNumber field.
4. Click the Save icon.



The new record has been saved to the database.

► **To retrieve the record you just entered**

1. Click the Search mode icon.



2. Type 100 in the OrderNumber field.
3. Click the Search icon.

The record you just entered appears in the window.

Closing the Window

When you are done experimenting with the CustomerOrder window, choose the File > Close command.

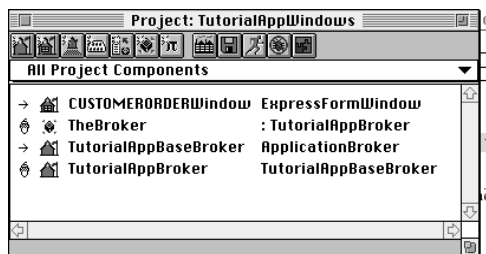
Viewing Generated TOOL Classes

Now that you have seen what a running window looks like, take a look at the generated CustomerOrderWindow class in the Window Workshop.

► **To view the CustomerOrderWindow in the Window Workshop**

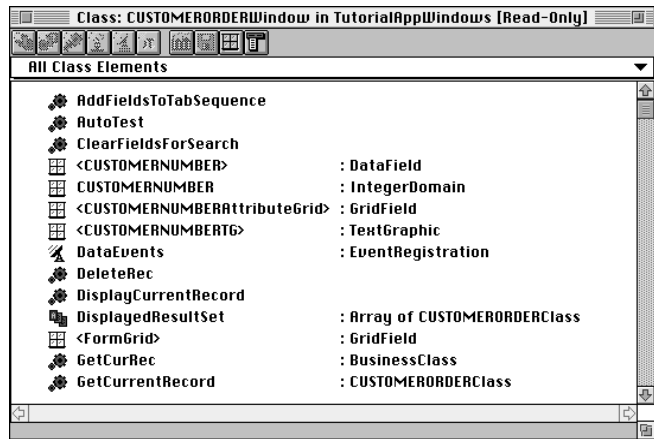
1. Return to the Repository Workshop.
2. Double-click TutorialAppWindows.

This opens the TutorialAppWindows project window.



3. Double-click CUSTOMERORDERWindow.

This opens the Class Workshop.

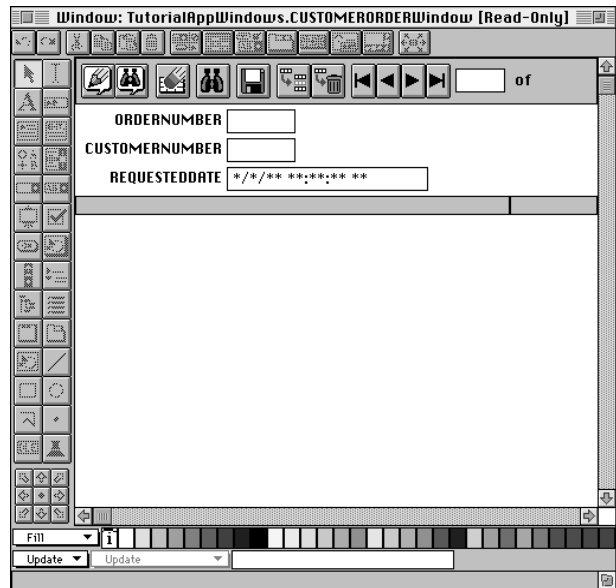


The title indicates that CustomerOrderWindow is a non-customizable class with the words "[Read-Only]."

4. Click the Window Workshop icon.



The figure below displays the generated window for class CustomerOrderWindow.



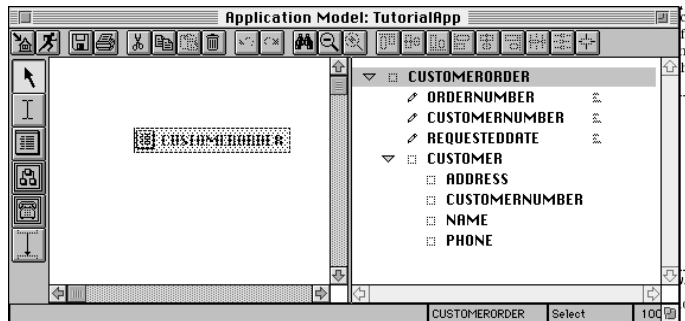
5. Close the Window Workshop.

Creating a Simple Window Based on Two Tables

Now that you have seen how easy it is to create a simple application, let's add to it. The current window is based on one table, CustomerOrder. In this next part of the tutorial, you will add some customer information to the window.

You can take advantage of the relationship between Customer and CustomerOrder in the business model as you build your application. The TOOL code generated from the business model includes a CustomerOrder class that contains a reference to the Customer class. This means that CustomerOrder can reference each attribute on the Customer class, which is reflected in the Attribute List in the Application Model Workshop. **Figure 3-3** shows the expanded Attribute list—notice the Customer attribute available as well as the CustomerOrder attributes.

Figure 3-3 Customer Attributes in CustomerOrder Attribute List



The following section describes how to place Customer fields on the CustomerOrder window.

Adding Fields to a Window

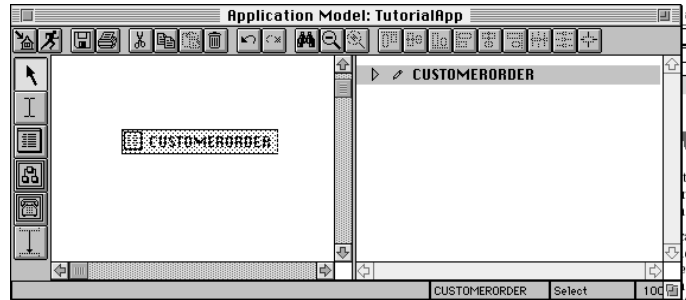
By default, all the CustomerOrder attributes appear as fields on the CustomerOrder window. To add customer information to the CustomerOrder window, you simply select the additional attributes you want. For example, you might want to display all the columns from the CustomerOrder table, as well as the customer name and address from the Customer table.

In most cases, any Customer attributes that appear on the CustomerOrder window will be read only. Simple business class windows allow you to modify data in only one business class at a time. (There is an exception to this rule in the case of an aggregate association. See [“Aggregation” on page 121](#) for details.) Therefore, because the CustomerOrder window is based on the CustomerOrder business class, you can only update the CustomerOrder table—the Customer information will not be editable.

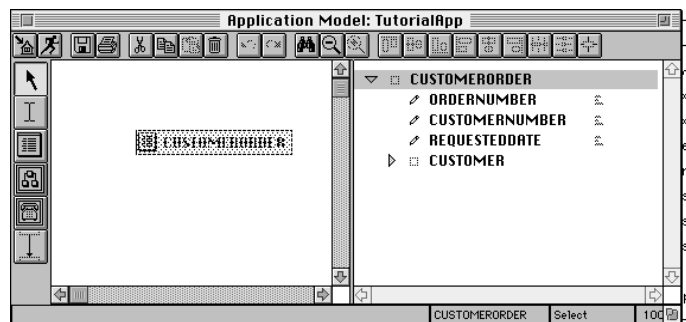
► To add Customer attributes to the CustomerOrder window

1. Select the CustomerOrder window.

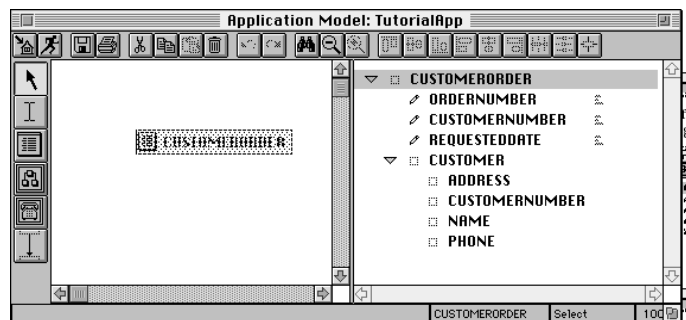
Notice CustomerOrder in the Attribute List, as shown below.



2. Click the expansion arrow to the left of CustomerOrder to display all the CustomerOrder attributes.



3. Click the expansion arrow to the left of Customer to display all the attributes from the Customer class, as shown in the figure below.

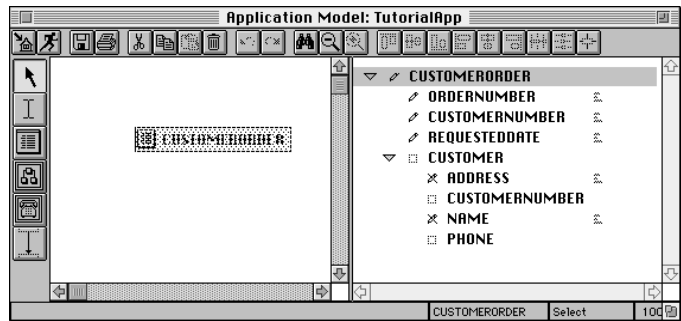


4. Click the box next to the Name attribute (under Customer).

A pencil with a line through it appears next to the Name attribute, meaning that the attribute is read only.

5. Click the box next to the Address attribute.

A pencil appears with a line through it.

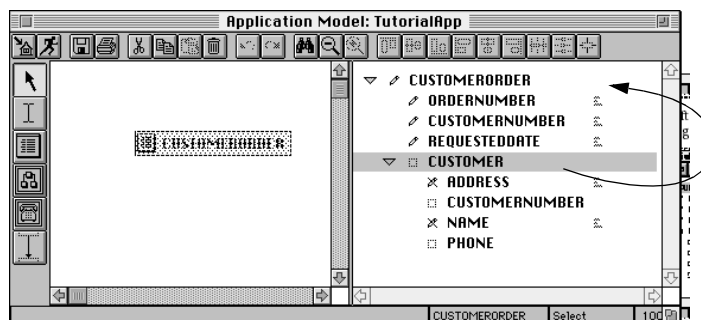


Modifying Field Order

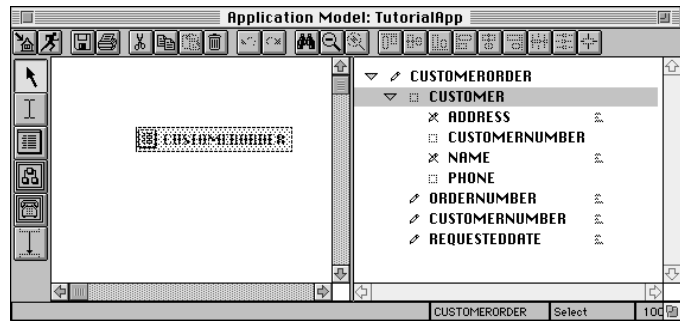
The order in which the attributes appear in the Attribute list determines the order the fields appear on the window. Thus, by default, all the CustomerOrder fields will appear before the Customer fields. For the CustomerOrder window, though, it may be useful to have the customer name and address appear above the order information. To rearrange the fields, you must rearrange the attributes from the Attribute list.

► To change field order

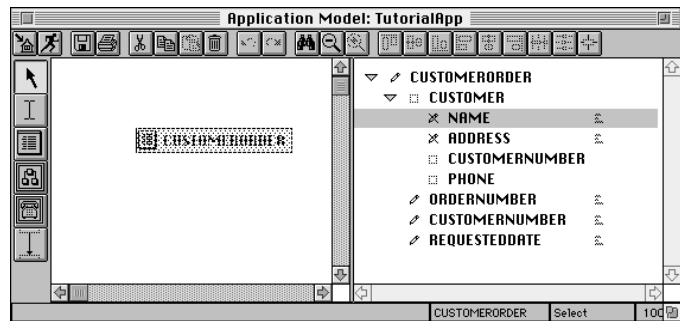
1. Drag the Customer row to a position just below the CustomerOrder field, as shown below.



All the Customer attributes appear above the CustomerOrder attributes, as shown below.



2. Drag the Name attribute above the Address attribute, as shown in the figure below.

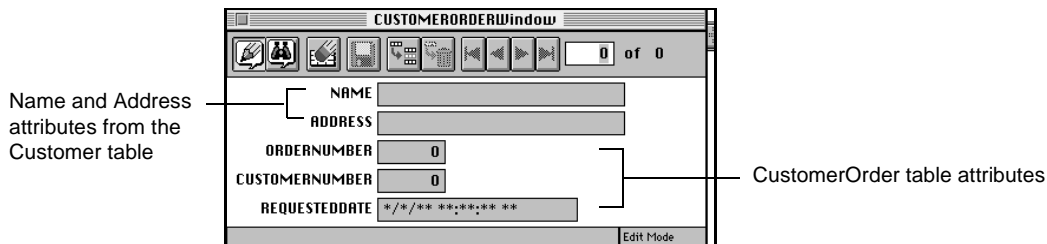


► **To run the window**

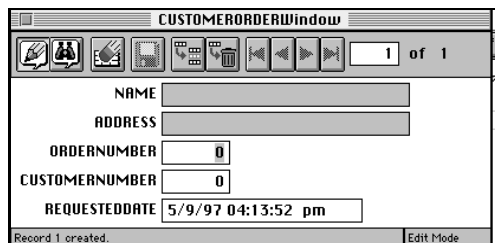
1. Select the Run Application command.

The Run Application command automatically generates code that reflects the changes to the attributes and runs the window.

A new window appears, as shown below.



If you select the Insert Record icon, the CustomerOrder attributes become text fields with a white background, which indicates an editable field, as shown below.



The attributes from the Customer table appear in a text field with a grey background, meaning they are read only. You can type search criteria into the Customer fields when the application is in Search mode

2. Choose the File > Exit All command to exit the application.

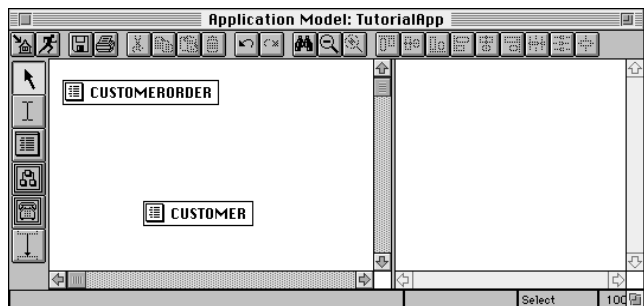
Using Multiple Windows

Thus far, you have an application that consists of only one window. In this section, you will add another window to the application.

The CustomerOrder window allows a user to update the CustomerOrder table. The customer fields on the CustomerOrder window, however, are read only, so you cannot insert the customer information. To get access to the Customer table, you are going to create a window based on the Customer table, and then specify a way to access this window from the CustomerOrder window.

- **To create a window based on the Customer business class**
 1. Drag the Customer class from the business class list in the Business Model Workshop into the drawing area of the Application Model Workshop.

A Customer business class window appears.



Next you will incorporate the Customer window into the application using a link.

About Links

A *link* is the way one window calls, or invokes, another window. Creating a link is similar to creating an association: you connect two window classes and define properties of the connection. You set link properties in the Link Properties dialog, which you open by double-clicking on the link (you can also select the link and choose the Component > Properties... command). This tutorial briefly covers link properties—refer to [Chapter 6, “Using the Application Model Workshop,”](#) for more information.

Creating a Lookup Link

You will use the Customer window to look up an existing customer. After selecting a customer, you will close the window, and Express sends the customer data back to the CustomerOrder window, where it will be used to populate the name, address, and customer number fields. You create this functionality using a lookup link.

► **To create a lookup link**

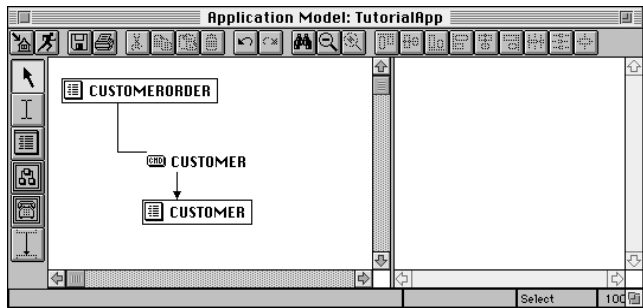
1. Select the New Link tool.



2. Drag the cursor from the CustomerOrder window icon to the Customer window icon.

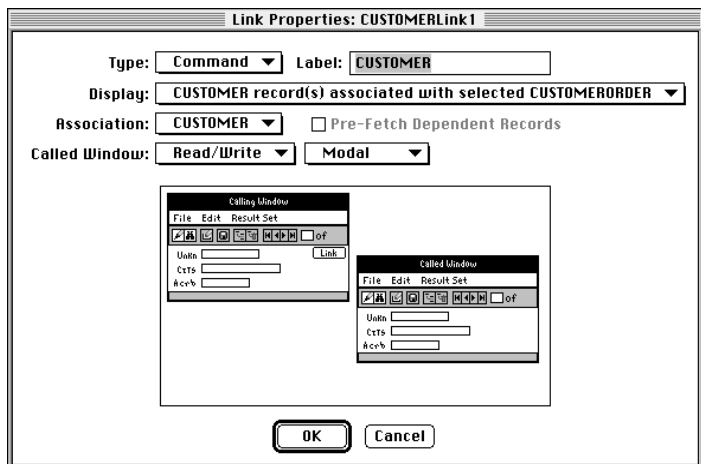
The direction of the link is important; start at the From class and drag to the To class.

A line appears with a command link icon, as shown below.



By default, Express creates a command link. To define a lookup link, you will use the Link Properties dialog.

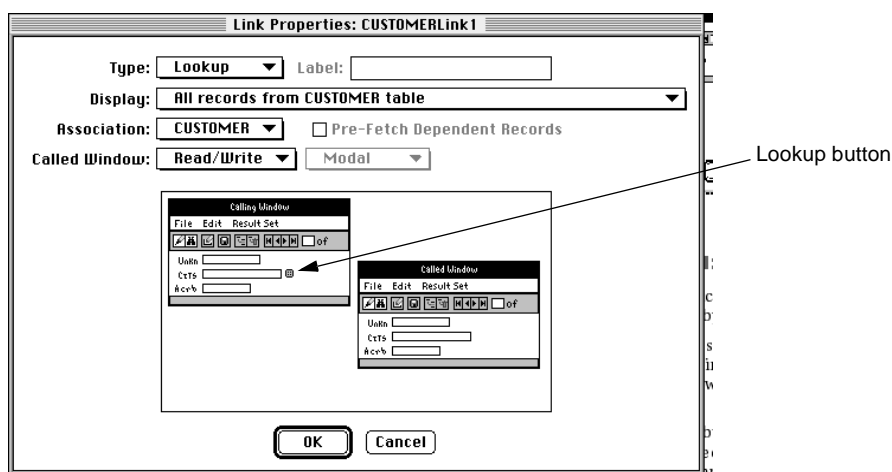
3. Double-click on the command link. The Link Properties dialog appears.



4. Select Lookup from the Type drop list.

When you create a lookup link, Express automatically assigns the Modal property to the window called by the link. In other words, once you invoke the Customer window, the CustomerOrder window is unavailable until you dismiss the Customer window. Dismissal takes the form of either selecting the Cancel button to cancel the lookup operation, or selecting the record and then selecting the OK button to close the window.

The Modal property cannot be changed in the dialog. This property is reflected in the Link Properties dialog, as shown below.



5. Click OK.

Modifying the Customer Window Properties

To provide the best interface for the lookup functionality, you will now make some changes to the Customer window by modifying its properties.

You modify business class window properties in the Business Class Window Properties dialog. You access the Business Class Window Properties dialog by double-clicking on the window, or selecting the window and choosing the Properties... command from the Component menu.

One of the properties of a business class window is the Command Set. The Command Set property allows you to specify what type of commands to include on your generated window. The default Command Set is All—this means that the following commands are included on the window:

- Search Mode

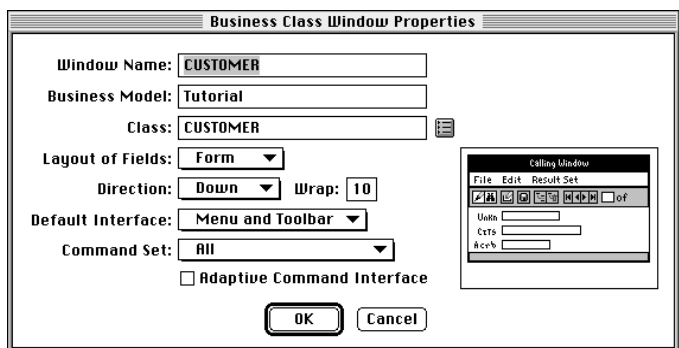
- Edit Mode
- Search
- Insert Record
- Delete Record
- Save
- Clear Result Set
- Scrolling commands

Because the Customer window will be used only to retrieve data, you will want to use the Search command set rather than the All command set that is the default. The Search command set supplies only the search and scrolling commands.

Another property on a business class window is Default Interface. The Default Interface property allows you to choose what type of command interface to use on the generated window. The types of interfaces are: Buttons, Menu and Toolbar. The Menu and Toolbar option is the default. Because you will always be required to close the Customer window before returning to the CustomerOrder window, it will be useful to have an OK button that closes the window. Dialogs such as this often display commands as buttons.

➤ **To apply the Search command set and the Buttons default interface**

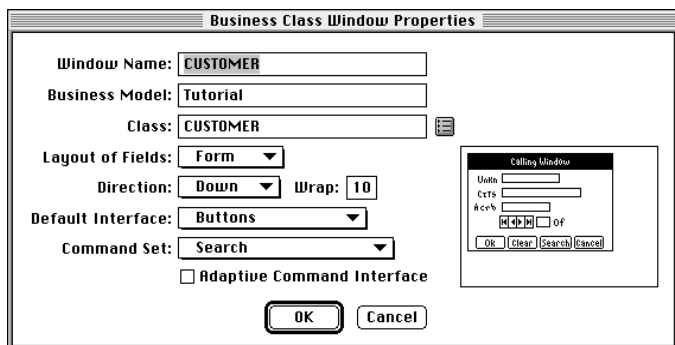
1. In the Application Model Workshop, double-click the Customer window to display the Business Class Window Properties dialog, as shown below.



2. Select Search from the Command Set drop list.

3. Select Buttons from the Default Interface drop list.

Your screen should look like the one below:



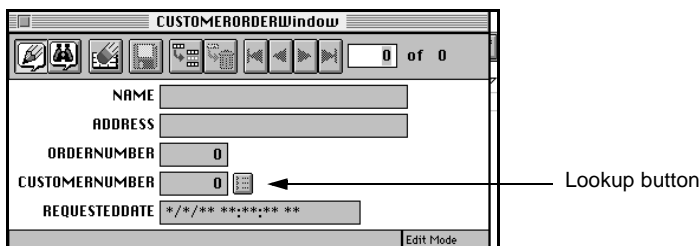
4. Click OK.

You have fully specified the lookup link and defined the Customer window. You can run the Customer window by clicking the Run Application icon.

Using the CustomerOrder Window

The CustomerOrder window is shown in Figure 3-4. Notice the lookup button.

Figure 3-4 CustomerOrder Window with Lookup Link to the Customer Window



You use the lookup window when you wish to enter a new order. For example, an existing customer calls, but does not know his or her customer number.

► **To use the lookup link to insert a record for an existing customer**

1. Click the Insert Record icon




The CustomerOrder fields become editable, and the Customer fields remain grey and uneditable, as shown below.

CUSTOMERORDERWindow

NAME

ADDRESS

ORDERNUMBER

CUSTOMERNUMBER 

REQUESTEDDATE

Record 1 created. Edit Mode

2. Click the lookup button.

All the records in the Customer table are fetched from the database when the window opens, as shown below.

CUSTOMERWindow

CUSTOMERNUMBER

NAME

ADDRESS

PHONE

 1 of 3


3. Scroll through the records until the desired record is displayed.
4. Click OK. The displayed record is returned to the CustomerOrder window, as shown below.

CUSTOMERORDERWindow

NAME

ADDRESS

ORDERNUMBER

CUSTOMERNUMBER 

REQUESTEDDATE

Record 1 created. Edit Mode

5. Complete the order by adding the order number and requested date.
6. Click the Save icon.



7. Choose the File > Exit All command to exit the application.

Enhancing the Application

Up to now, you have learned about some fairly basic features of Express. In this section, you will add some more features to your application, taking advantage of the power of Express.

Creating a Nested Link

A nested link creates a relationship between two windows. When you use a nested link, the called window appears within the calling window, and the called window displays data based on information from the calling window.

For example, imagine your order entry system. You have already created a window that displays orders. It would be useful to create a window that displays all the line items that belong to a specific order. You do not want to create a window that displays only line items, because line items are meaningless without the orders to which they belong. (Remember, the association in the business model between CustomerOrder and LineItems is an aggregation.)

This situation is a perfect place to use a nested link. The user will want to see only line item information that relates to a specific order, and would like to view all of the information in one window.

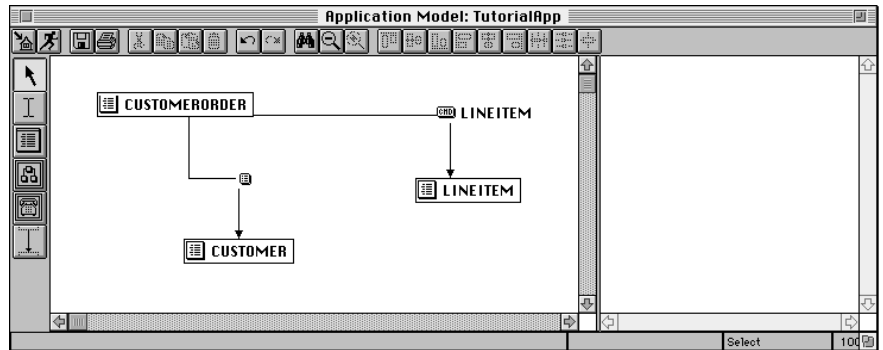
► To create a nested link

1. In the Business Model Workshop, drag the LineItem class from the Business Class List into the Application Model Workshop.

A LineItem business class window appears.

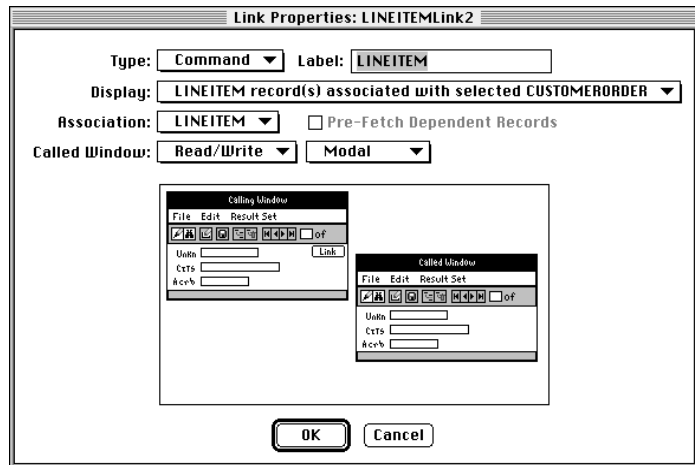
2. In the Application Model workshop, select the Link tool and draw a link from CustomerOrder to LineItem.

The application model should now look like the figure below.



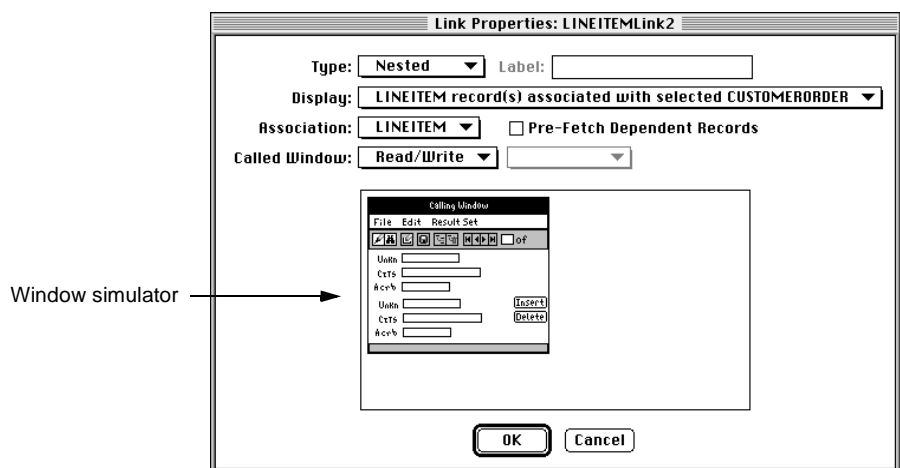
3. Double-click on the LineItem link.

The Link Properties dialog appears, as shown below.



4. Select Nested from the Type drop list.

Notice that the Mode status property becomes unavailable, and the window simulator changes to display a nested window. The following figure illustrates.



5. Click OK.

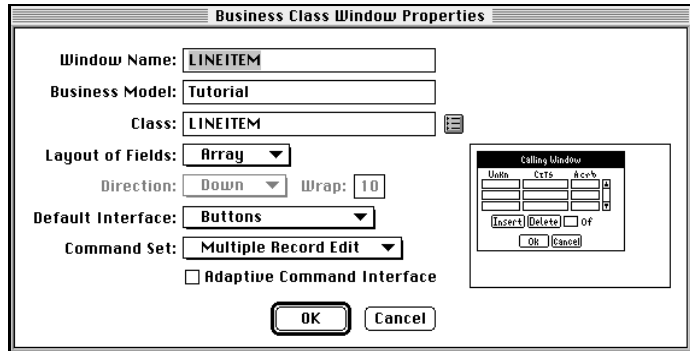
Since you will want to view all the line items associated with an order, an array window will be the best format for the LineItem window. Also, you do not need the order number to appear again with the line items, because it already appears with the order information. Finally, to be able to insert and delete line item records, you need to choose the Buttons default interface. The Buttons interface provides insert and delete push-button that act on the nested portion of the window.

6. Double-click on the LineItem window to display the Business Class Window Properties dialog.
7. Select Array from the Layout of Fields drop list.

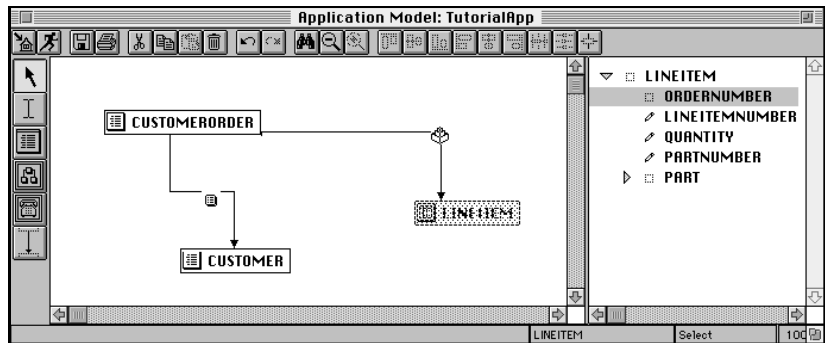
Notice that the Window Simulator now depicts an array window.

8. Select Buttons from the Default Interface droplist.

Your screen should look like the one below.

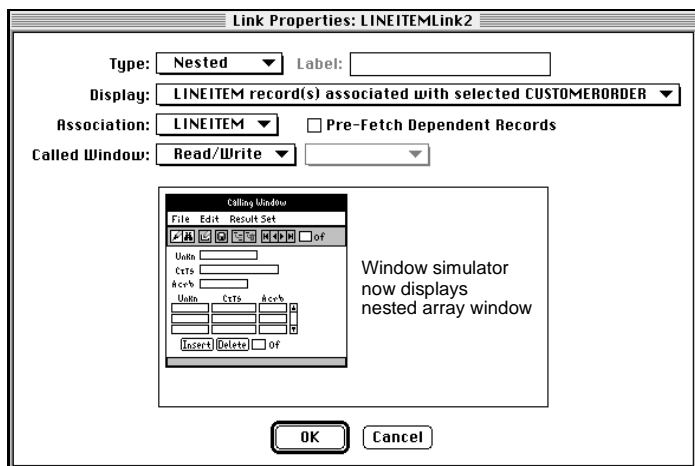


9. Click OK.
10. In the Attribute List, deselect the OrderNumber attribute by clicking on the icon to the left of it until the icon becomes an empty box, as shown below.



- Double-click on the LineItem link to bring up its properties dialog.

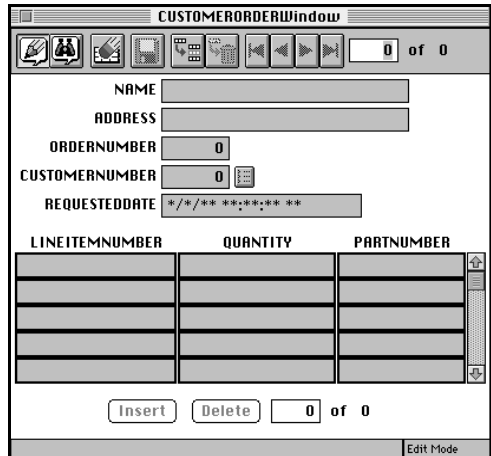
Note that the Window Simulator in the nested link's Link Properties dialog now displays a nested array window, rather than the nested form window shown below.



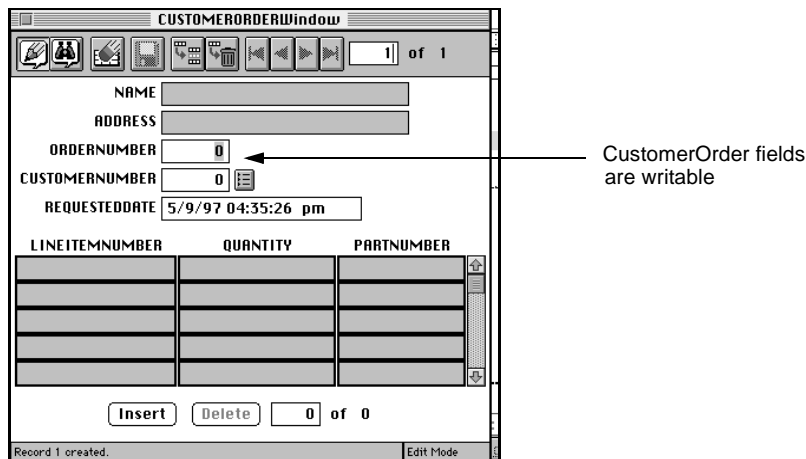
Using the CustomerOrder Window

Now click the Run Application icon to run the modified CustomerOrder window. The nested Lineltem window appears in the CustomerOrder window, as shown in [Figure 3-5](#).

Figure 3-5 CustomerOrder Window with Nested Lineltem Window



- **To enter a new order for an existing customer**
 1. Click the Insert Record icon.



2. Click the lookup button.

The Customer lookup window appears, as shown in the figure below.

3. Scroll to a record in the Customer window and click OK.

The Customer record is returned to the CustomerOrder window, as shown in the figure below.

LINEITEMNUMBER	QUANTITY	PARTNUMBER

4. Type an Order Number.
5. Click the Insert button.

6. In the array, type a line item number, quantity, and part number.
Your window should look like the figure below.

The screenshot shows a window titled "CUSTOMERORDERWindow" with a toolbar at the top containing icons for search, save, delete, and navigation. The main area contains a form with the following fields:

- NAME: [Empty text box]
- ADDRESS: [Empty text box]
- ORDERNUMBER: [Text box containing "2"]
- CUSTOMERNUMBER: [Text box containing "1"]
- REQUESTEDDATE: [Text box containing "5/9/97 04:47:29 pm"]

Below the form is a table with the following structure:

LINEITEMNUMBER	QUANTITY	PARTNUMBER
1	25	456

At the bottom of the window, there are buttons for "Insert", "Delete", and "1 of 1". The status bar at the very bottom indicates "Record 2 created." and "Edit Mode".

7. Click the Save icon in the menu bar.
- **To retrieve a customer order**
 1. Click the Search Mode icon.
 2. In the OrderNumber field, type the Order number you just used for the new record.
 3. Click the Search icon.
 - **To insert a new line item on an existing order**
 1. Click the Insert button.
 2. Type in the new line item information.
 3. Click the Save icon in the toolbar.
 4. Exit the application by choosing the File > Exit All command.

Customizing

There is a wide variety of customizations you can perform on Express applications. The following sections describe two simple ways to customize the Tutorial application:

- changing the format of a data entry field
The specific change applies a format template to the Requested Date field to display dates in a simpler format.
- changing the default mode of a window
Changes the default mode of the application's start window from Edit to Search.

Business and window classes that you generate are not customizable. To customize classes, you must first create customizable subclasses of the base classes. To customize individual classes, which is what you will do for these examples, you use the Customization Manager. The Customization Manager and other ways to create customizable classes are fully described in the *Customizing Express Applications* manual.

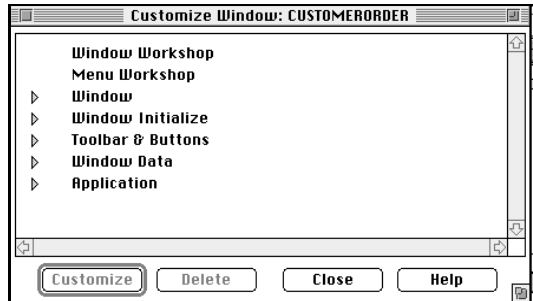
Changing the Format of a Data Entry Field

In the current Tutorial application, the date format of the Requested Date field displays not only the day, month, and year, but also hours, minutes, seconds, and am or pm. Such precision of time is not always necessary for entering orders of merchandise. To change the requested date field to a simpler format requires that you apply a different format template to the Request Date field. To do this, you will first create a customizable subclass of the CustomerOrder window.

► **To apply a template to a field**

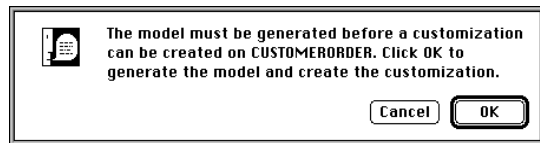
1. In the Application Workshop, select the CustomerOrder class.
2. Choose the Component > Customize... command.

The Customization Manager appears.



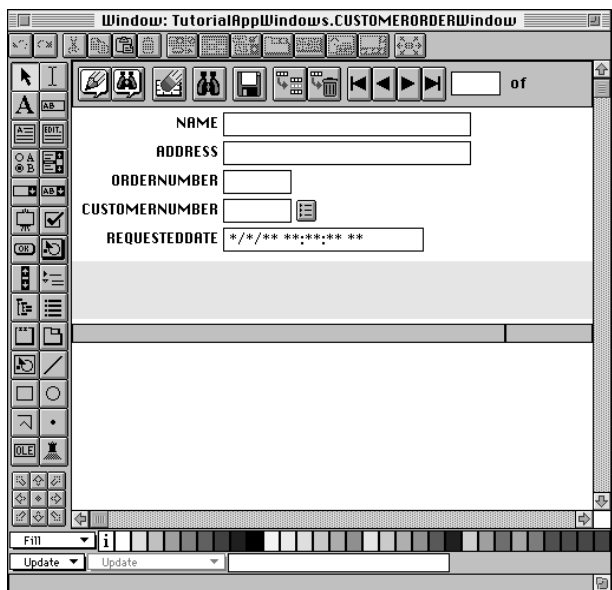
3. Double-click Window Workshop, or select Window Workshop and click Customize.

An alert window appears:

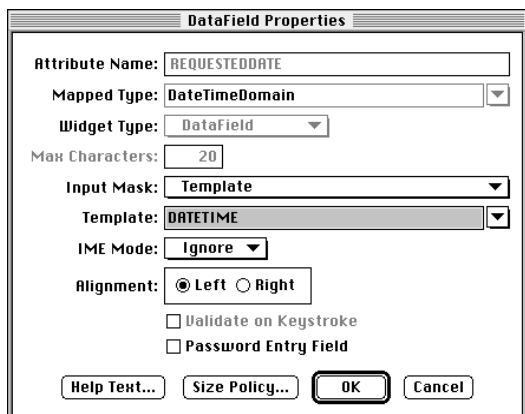


4. Click OK.

Express automatically expands the class hierarchy to include a customizable class for the CustomerOrder window, and the Window Workshop opens, displaying an editable version of the CustomerOrder window.

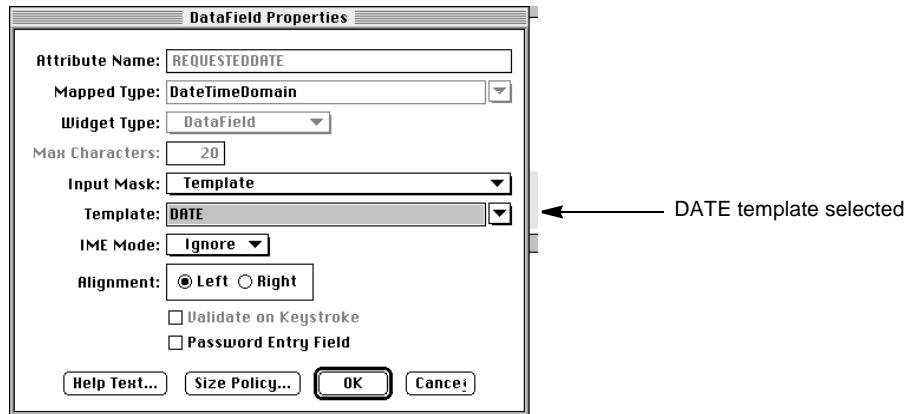


5. Double-click on the REQUESTEDDATE field to bring up its property dialog.



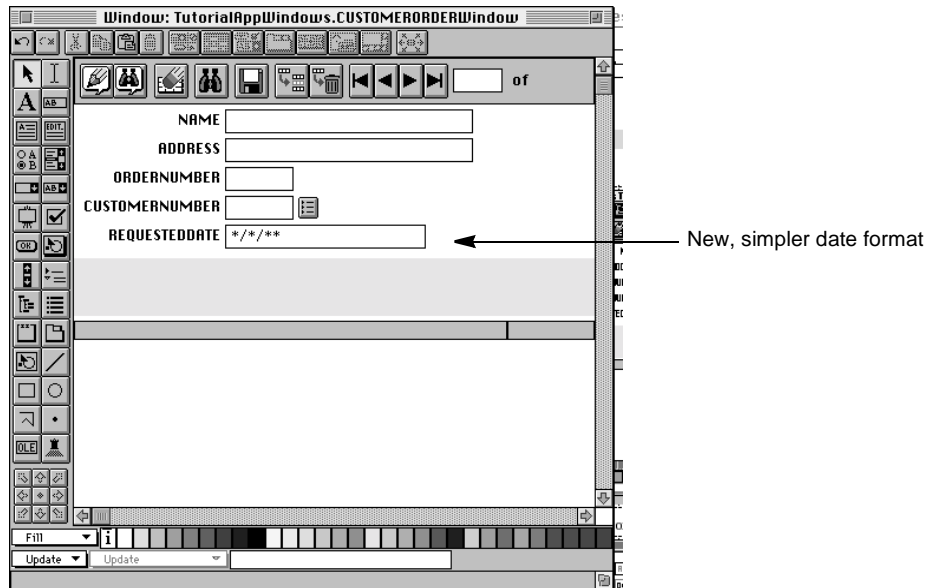
6. Select DATE from the Template list of formats.

Your datafield property dialog should look like this:



7. Click OK.

The CustomerOrder window shows the change in the date format.



8. Close the Window Workshop and the Customization Manager.

Now when you run the example application, dates in the Requested Date field always appear in the new date format.

NOTE Records that you inserted before the customization still have time data associated with them in the database. When you want to query such records by date, you must take this into account. For information on such queries, see [“Retrieving Data” on page 241](#).

Remember, if you run the window from the Repository or Project Workshop, you must regenerate the classes in the Application Model Workshop. If you run from the Application Model Workshop, Express regenerates the classes automatically.

Viewing the Generated Customizable Class

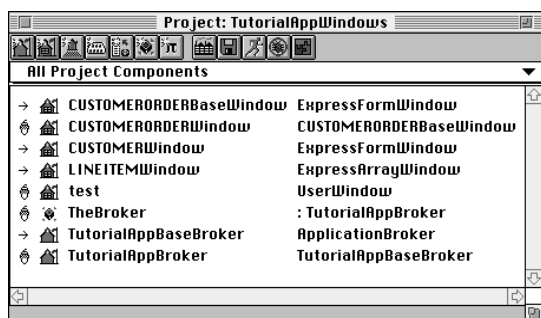
As mentioned in the introduction to this section, running the Customization Manager on a class creates a new customizable subclass of the class generated for the application. As shown in [“Viewing Generated TOOL Classes” on page 59](#), the CUSTOMERORDERWindow class is one of the classes generated for this tutorial application. It is shown in that section as a subclass of the ExpressFormWindow class.

Now that you have created a customizable class, take a look at the generated classes for the Tutorial application.

► To view the generated TutorialApp classes

1. Return to the Repository Workshop.
2. Double-click TutorialAppWindows.

This opens the TutorialAppWindows project window.



Note that CUSTOMERORDERWindow is now a subclass of a new class, CUSTOMERORDERBaseWindow, which is a subclass of ExpressFormWindow. What happened?

In the process of creating a customizable class, the Customization Manager renames the original class (CUSTOMERORDERWindow, in this case) by inserting "Base" into the name (thus, CUSTOMERORDERBaseWindow), then creates a new, customizable class, which it names with the original class's name (CUSTOMERORDERWindow).

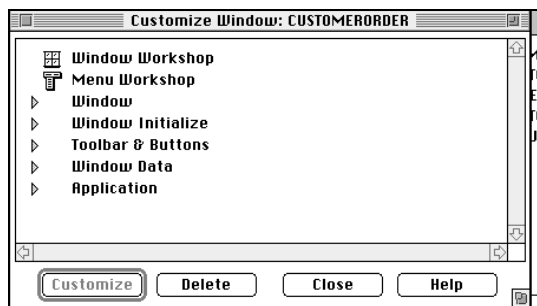
Changing the Default Mode of a Window

In the current Tutorial application, the CustomerOrder window is in edit mode when you start the application. In this customization, we change it to start in search mode.

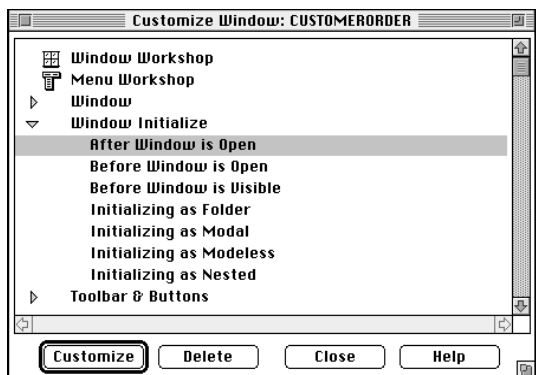
► To change the default mode of a window

1. In the Application Workshop, select the CustomerOrder class.
2. Choose the Component > Customize... command.

The Customization Manager appears.

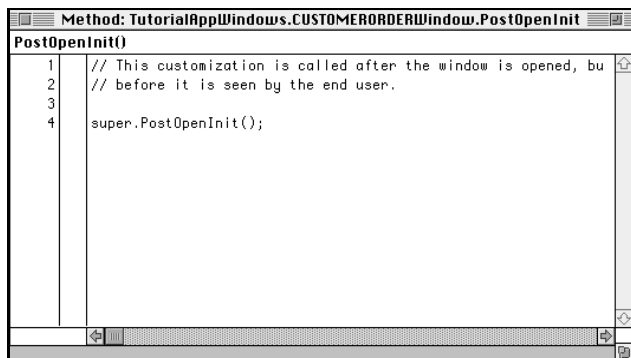


3. Click on the Window Initialize triangle to open it.



4. Double-click on After Window Is Open.

The Method Workshop appears.



5. Type the following code as the last line in the method body:

```
self.SetSearchMode(confirm = FALSE);
```

- Choose the File > Compile command to compile the method, which determines whether there are errors.

Your Method Workshop should look like this:

The screenshot shows a window titled "Method: TutorialAppWindows.CUSTOMERORDERWindow.PostOpenInit". The window contains a code editor with the following text:

```

PostOpenInit()
1 // This customization is called after the window is opened, bu
2 // before it is seen by the end user.
3
4 super.PostOpenInit();
5 self.SetSearchMode(confirm = FALSE);
    
```

The code editor has a line number column on the left and a scroll bar on the right. The window also has standard window controls (minimize, maximize, close) in the top right corner.

- Close the Method Workshop and the Customization Manager window.

Now when you run the example application, the CustomerOrder window appears in search mode by default. And once again, if you run the window from the Repository or Project Workshop, you must regenerate the classes in the Application Model Workshop. If you run from the Application Model Workshop, Express regenerates the classes automatically.

Using the Repository Workshop

This chapter provides information about using the iPlanet UDS Repository Workshop when you are working with Express.

In this chapter, you will learn how to:

- check out and branch Express models
- use the Compile All Plans command with Express
- set Repository Workshop preferences for Express

Checking Out and Branching Models

To get write access to an existing business or application model, you must either check out or branch the entire plan. The Repository Workshop provides Checkout and Branch commands that allow you to check out or branch a business or application model. The following two sections describe how to check out and branch models. For complete information on using the Repository Workshop, see *A Guide to the iPlanet UDS Workshops*.

To get write access to a project, you must check out or branch individual project components using the Project Workshop. You cannot check out or branch an entire project using the Repository Workshop.

The Repository workshop also provides commands that let you revert the changes you made to a model that you have checked out or branched or to undelete a model that you deleted after checking it out. These commands are described under [“How to Undo Changes” on page 92](#).

Checking Out a Plan

The Checkout command in the Repository Workshop gives you an exclusive write lock on a business or application model. Before you can check out a model, the workspace must be open for modifying. This means no other workspace can currently have the model checked out, and your workspace must contain the latest version of the model.

After you have finished your modifications, you do not have to check the model back in. It is automatically checked in when you integrate your workspace.

► To check out a business or application model

1. In the Plan browser, select the business or application model you wish to check out.
2. Choose the Plan > Checkout command.
3. When the model is checked out, the Plan browser displays a checkout icon by the model name.

If the model you check out is branched, the branch will be converted to a checkout. This will work only if no other workspace has integrated the model since you last performed an Update Workspace command on the workspace, or ran an Include Public Plan command on that model.

The Checkout command will fail if the model is already checked out by another workspace. The error message will give the name of the workspace that has checked out the model. It will also fail if the workspace does not have the latest version of the model. In this case, use the Update Workspace command to bring the latest version of the model into your workspace.

If you wish to undo the changes you have made since the Checkout command, and revert the model to the state it was in after your last Update Workspace command, you can use the Undo Checkout/Branch command as described below.

Branching a Plan

The Plan > Branch command gives you temporary write access to a business or application model in order to test out a change while someone else has checked out the model. You cannot integrate changes in a branched model into the system baseline.

Before you can branch a model, the workspace must be open for modifying. Any number of workspaces can branch a model at the same time.

► To branch a business or application model

1. In the Plan browser, select the business or application model you wish to branch.
2. Choose the Plan > Branch command.
3. When the model is branched, the browser displays a branch icon by the model name.

If the model you branch is checked out, the checkout will be converted to a branch. When a checkout is converted to a branch, your changes to the model are retained in the workspace, but the model is free to be checked out by another workspace after the next Save All command.

If you wish to undo the changes you have made since the Branch command, and revert the model to the state it was in before your last Branch command, you can use the Undo Checkout/Branch command as described next.

How to Undo Changes

The Plan > Undo submenu provides two commands that let you undo changes.

The Undo > Checkout/Branch command lets you erase all changes you made to the model since you gave the Checkout or Branch command. The model reverts to the state it was in before the Checkout or Branch command, and the model is freed for checkout by another workspace.

➤ **To revert a model**

1. In the Plan browser, select the model you wish to revert.
2. Choose the Plan> Undo command.
3. On the Undo command's submenu, choose the Checkout/Branch command.

The Undo > Deleted Checkouts command "undeletes" a model that you deleted after having checked it out.

➤ **To undelete a model**

1. Choose the Plan > Undo command.
2. On the Undo command's submenu, choose the Deleted Checkouts... command.
3. In the Deleted Checkouts dialog, select the plan you wish to undelete.
4. Click the OK button to undelete the component.

The model you have "undeleted" will revert to the state it was in before you originally checked it out. It will appear in the Plan browser, and it will no longer be checked out.

Of course, if you have deleted a model that was never integrated, you cannot use the Undo Deleted Checkouts... command to undelete it.

Compiling Plans

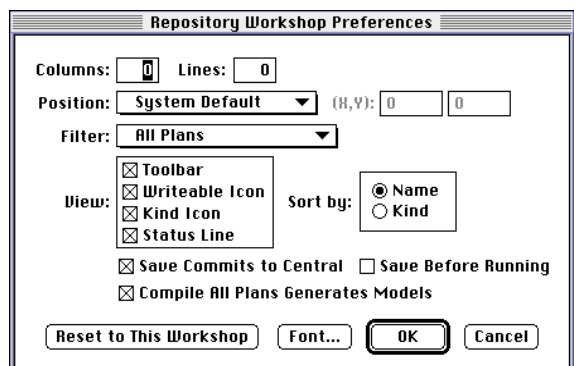
The File > Compile All Plans command compiles all out-of-date plans. If the Compile All Plans Generates Models toggle is checked in the Repository Workshop Preferences dialog, then all out-of-date business and application models are also regenerated.

The submenu of the Utility > Force Compile command lists All Plans and Selected Plan. If a plan is a model, the Utility > Force Compile > Selected Plan command generates the code from the selected model. The Utility > Force Compile > All Plans command generates models if the Compile All Plans Generates Models preference has been set in the Repository Workshop Preferences dialog. For information on setting this preference, see [“Repository Workshop Preferences” on page 93](#).

Repository Workshop Preferences

For users who have Express installed, there is a workshop preference for the Repository Workshop called Compile All Plans Generates Models, as shown in [Figure 4-1](#). If you want the File > Compile All Plans command to regenerate all out-of-date Express models, as well as out-of-date projects, check the Compile All Plans Generates Models toggle. See [“Generating Server Code” on page 143](#) for information about regenerating all out-of-date Express business models, and [“Generating Client Code” on page 213](#) for regenerating out-of-date application models.

Figure 4-1 Repository Workshop Preferences Dialog



Using the Business Model Workshop

The first step in creating an application is to develop a business model. A business model represents all the objects in a business application and the relationships between them. The Business Model Workshop provides tools for drawing and defining a business model. This chapter covers the following topics:

- using the toolbar and palette
- setting business model properties
- drawing business model elements
- creating relationships between business classes
- creating services
- generating server classes
- setting custom generation options
- importing database schema
- exporting object models
- setting workshop preferences
- command summary

About Business Models

A business model is the foundation of an application you build with Express. Although you are not required to have a complete model before you begin the development of your application model, you will ultimately need a complete model of your system to implement all the functionality you desire.

Business models are comprised of business classes, attributes, and relationships between classes. Each business class is based on a database table. Columns in the table correspond to attributes in the business class.

If you are building an application that will run on existing corporate data, you can jump-start your business model by importing the database schema. iPlanet UDS will create a business class for each database table. Conversely, you might be designing your database along with your application. In this case, you can diagram your business model by drawing classes and attributes and then use an export feature to build a script containing the appropriate create table statements to build the database schema. See [“Importing and Exporting Database Schema” on page 135](#) for more information.

If you are modeling a large system, you can create multiple business models to accommodate all the tables. You can use multiple business models as suppliers to one application model.

Once you have created some or all of your business classes, you can use the Application Model Workshop (see [Chapter 6, “Using the Application Model Workshop”](#)) to build the client side of your application.

Using the Business Model Workshop

You enter the Business Model Workshop from the Repository Workshop. You can either create a new business model or edit an existing model. If you wish to edit an existing model, you will need write access to it. See [“Accessing Existing Business Models” on page 97](#) for information about checking out and branching a business model.

Creating a New Business Model

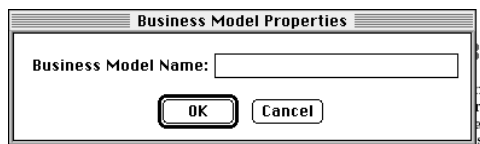
Creating a new business model involves opening the Business Model Workshop and naming a new model.

➤ **To create a new business model**

1. In the Repository Workshop, click the New Business Model icon, or choose the Plan > New Business Model... command.



The Business Model Properties dialog appears, as shown below:



2. Enter a business model name.

Business Model Name property The name you give the business model will be used to name the project generated from the business model. Express appends “Services” to the business model name to create the project name. For information on the project generated from the business model, see *Customizing Express Applications*.

3. Click OK.

The Business Model Workshop window appears and you can begin to create a business model. See [“The Business Model Workshop Window” on page 99](#) for a description of the workshop.

Accessing Existing Business Models

As discussed in [Chapter 4, “Using the Repository Workshop,”](#) to get write access to a business model that has been integrated into the repository, you must either check out or branch the entire plan. Note that because an entire model must be checked out, two people cannot work on a model at the same time. For this reason, it is often useful to break up a model into multiple business models, if possible.

► To modify an existing model in the Business Model Workshop

1. In the Repository Workshop, check out or branch the model using the appropriate command on the Plan menu.

See [“Checking Out and Branching Models” on page 89](#) for complete information.

2. In the Repository Workshop browser, double-click on a business model name.

The Business Model Workshop appears displaying the business model you selected.

Once you have opened the Business Model Workshop, you are ready to edit the existing business model.

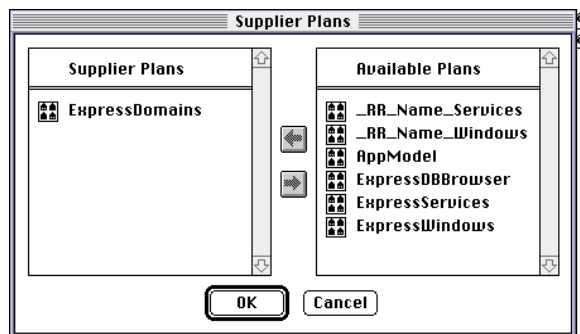
Adding Supplier Plans

You can add supplier plans to your business model by using the Supplier Plans command on the File menu; only projects are allowed as supplier plans to business models. Typically, you need to include a supplier project in your business model when you create custom domains (see [“Assigning Custom Domains” on page 116](#)). To be able to access the custom domains from your business model, the TOOL project that contains the custom domains must be a supplier plan to the business model.

► To add a supplier plan

1. Choose the File > Supplier Plans... command.

The Supplier Plans dialog appears, as shown below.



2. Under Available Plans, select the plan that you wish to add as a supplier and click the left arrow.

The plan you selected now appears under Supplier Plans.

3. Repeat until you have transferred all the necessary plans.
4. Click OK.

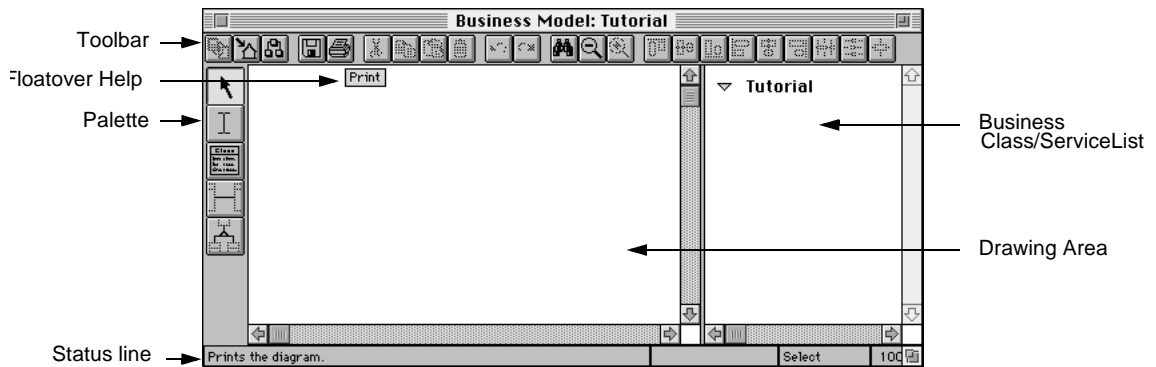
Supplier Plans and Generated Projects

If you add supplier plans to a business model, Express automatically adds the plan as a supplier to the generated project when the project is generated. However, if you remove the supplier plan from the business model, you must manually remove the plan as a supplier to the generated project.

The Business Model Workshop Window

The Business Model Workshop, shown below, provides the tools you need to create and edit a business model.

Figure 5-1 Business Model Workshop Window



The Business Model Workshop window contains the following elements:

Toolbar The toolbar provides a set of tools for editing and aligning graphical elements of the business model.

Palette The palette allows you to create and manipulate a business model. In addition to the text tool and the select tool, the palette provides a subset of symbols that represent the Object Modeling Technique. These symbols are described in [“Using the Palette” on page 100](#).

Floatover Help Floatover help displaying the name of the component appears when you hold the cursor over any business model component.

Menus The menus in the Business Model Workshop provide commands that duplicate the functions of the toolbar and palette, as well as commands for additional features, such as printing.

Drawing area The drawing area is the space in which you create a business model.

Status Line The status line is divided into four sections displaying the following information:






- the first section displays the hint text for the icon over which the cursor is positioned

- the second section displays the name of the selected business model component
- the third section displays the tool mode, for example, select or edit
- the fourth section displays the current zoom percentage

Business Class/Service List This area displays the name of each business class you create or import. This area also displays the name of the default business service. See [“Business Class/Service List” on page 110](#) for more information.

Using the Palette

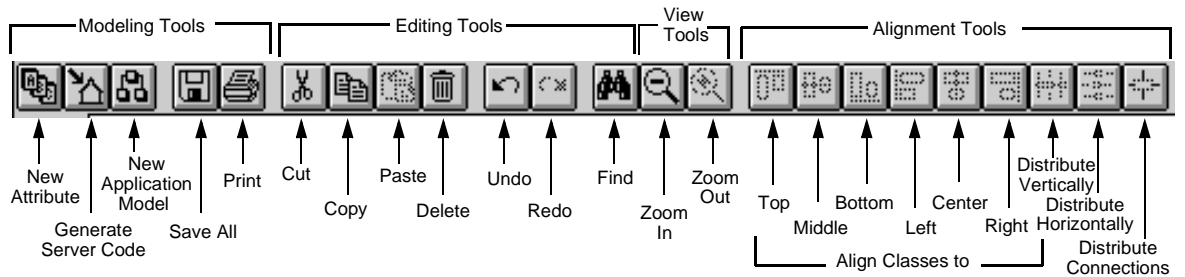
The Business Model Workshop palette provides tools for selecting, creating, and editing graphical elements.

Icon	Tool	Description
	Select Tool	Use the Select tool to select, move, and resize elements in the business model. Also use the Select tool to open the property dialog of any element of an object model by double-clicking on the element. Select multiple items by drawing a selection box around them or hold down the Shift key while selecting.
	Text Tool	Use the Text tool to enter or edit the name of a class, attribute, attribute domain, or association. When you activate the Text tool, the cursor is an arrow; when you move the cursor over text, it becomes an I-beam and you can enter, delete, or select text as you would in a word processing application.
	New Business Class Tool	Use the New Business Class tool to create a new business class.
	New Association Tool	Use the New Association tool to draw an association between classes.
	New Generalization Tool	Use the New Generalization tool to create a generalization between a class and its superclass.

Using the Toolbar



The Business Model Workshop toolbar provides a set of tools for editing and formatting a business model. Many tools are equivalents to the menu commands.




Figure 5-2 Business Model Workshop Toolbar Icons



Modeling Tools





Modeling tools are tools you use frequently in creating your model. Most modeling tools are equivalents of File menu commands. The New Attribute tool corresponds to a command on the Component menu.




Icon	Tool	Description
	New Attribute Tool	Creates a new attribute. Select the class on which you wish to define the attribute, click the New Attribute icon, and enter the appropriate information in the dialog. Refer to “Attributes” on page 112 for more information. Equivalent of the Component > New Attribute command.
	Generate Server Code Tool	Generates the TOOL classes that are represented by the business model. These classes are stored in the <i>business_modelServices</i> project. Equivalent of the File > Generate Server Code command.

Icon	Tool	Description
	New Application Model Tool	<p>Once you have developed part or all of your business model, you can begin constructing a corresponding application model. To create a new (empty) application model, click the New Application Model tool, or choose the File > New Application Model... command. If you select multiple business classes and then click the New Application Model tool, a default application model is generated containing business class windows for each selected business class.</p> <p>Equivalent of the File > New Application Model... command.</p>
	Save All Tool	<p>Saves all the changes you have made in your workspace.</p> <p>Equivalent of the File > Save All command.</p>
	Print Tool	<p>Prints a business model, including portions not currently being displayed. If the model is larger than a single page, the Print tool automatically tiles the diagram onto multiple pages, tiling left to right, top to bottom.</p> <p>Equivalent of the File > Print... command.</p>

Editing Tools

Editing tools allow you to manipulate the elements of your business model as you would in any graphical drawing program.

Icon	Tool	Description
	Cut Tool	<p>Removes the selected component and copies it to the clipboard.</p> <p>Equivalent of the Edit > Cut command.</p>
	Copy Tool	<p>Copies the selected component to the clipboard.</p> <p>Equivalent of the Edit > Copy command.</p>
	Paste Tool	<p>Pastes the contents of the clipboard to the location you select.</p> <p>Equivalent of the Edit > Paste command.</p>
	Delete Tool	<p>Deletes the selected component.</p> <p>Equivalent of the Edit > Delete command.</p>

Icon	Tool	Description
	Undo Tool	Undoes your most recent action. You can undo successive actions as far back as your first action in the workshop since your last save. Actions include entering information in property sheets. Equivalent of the Edit > Undo command.
	Redo Tool	Redoes your most recent action (cancels Undo). Equivalent of the Edit > Redo command.
	Find Tool	Allows you to search for a business class. See Equivalent of the Edit > Find... command.

In Select mode, you can cut or copy the following components:

- attributes (single or multiple)
- classes (single or multiple)
- associations (along with the classes they connect)
- generalizations (along with the classes they connect)

Note that you cannot cut and paste an association or generalization independently of the two classes that they connect. Instead, you must select both the classes and the association or generalization that connects them, before using the Cut tool; the same is true for the Copy tool.

You cannot manipulate services with the Cut, Copy, or Paste tools.

Using Cut, Copy, & Paste In Text Mode, you can cut or copy text from any part of the diagram onto the clipboard.

➤ **To cut, copy, or delete an element**

1. Select the element in your business model.
2. Click the Cut, Copy, or Delete tool.

The Paste tool copies the component in the clipboard to the diagram.

➤ **To paste an element**

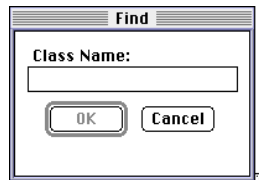
1. Cut or copy an element using the method described above.
2. Click the Paste tool.
3. Click on the position in the window where you want to paste the element.

Using the Find tool The Find tool is especially useful if your model is large and you cannot see all the classes in the drawing area at one time.

► **To find a business class**

1. Click the Find tool.

The Find dialog appears.





2. Enter the name of the class you wish to find and click OK.

Express highlights the business class (and brings it into view if it is not currently in the drawing area), and highlights the class name in the Business Class List.

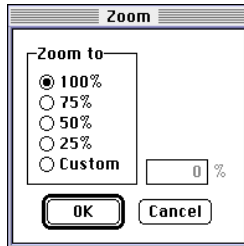
Viewing Tools

The View tools include the toolbar Zoom tools, the View > Zoom... command, and the View > Grid... command. The toolbar tools are shown in the following table; the command descriptions follow the table.

Icon	Tool	Description
	Zoom In Tool	Doubles the size of the current business model. This tool only affects a model that is being displayed at less than 100%. Equivalent of the View > Zoom In command.
	Zoom Out Tool	Reduces the size of the current business model. By default, each time you select this tool, the view reduces by 50%. Equivalent of the View > Zoom Out command.

Using the Zoom... command The Zoom... command does not have a toolbar equivalent. To zoom to a specific percentage view of the business model, choose the View > Zoom... command. The Zoom dialog appears, as shown in [Figure 5-3](#).

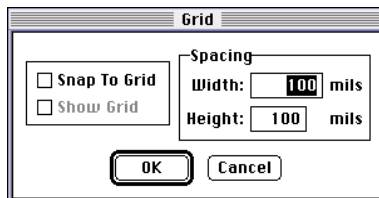
Figure 5-3 Zoom Dialog



Click one of the radio buttons to zoom to a preset percentage (100%, 75%, 50%, 25%) and then click OK. You can also click the Custom radio button and then enter a specific percentage in the field to the right and then click OK.

Using the Grid... command The Grid... command does not have a toolbar equivalent. You can modify alignment grid properties using the View > Grid... command. Choose the View > Grid... command and the Grid dialog appears, as shown in [Figure 5-4](#).

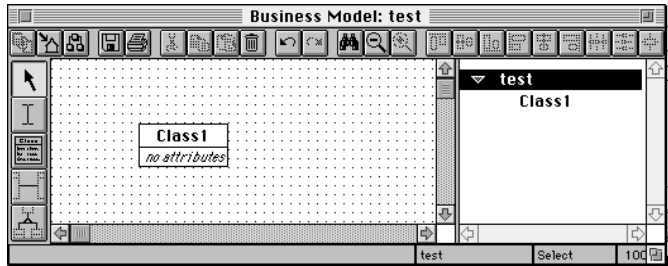
Figure 5-4 Grid Dialog



Snap To Grid property Turn on the Snap To Grid toggle to subsequently align business model components with grid lines. This property will affect only those components you create after you turn on the toggle.

Show Grid property The Show Grid property becomes active if you turn on the Snap to Grid property. Turn on the Show Grid property to display the grid lines in the drawing area, as shown [Figure 5-5](#).







Figure 5-5 Grid Lines






Spacing To specify the size of the grid, enter the Width and Height in mils.

Alignment Tools

The alignment tools help format business models.

Icon	Command	Description
	Align Classes to Top Tool	Aligns the tops of selected classes to the top-most class. Equivalent to the Arrange > Align Classes to Top command
	Align Classes to Middle Tool	Aligns the selected classes vertically through the middle of the classes. Equivalent to the Arrange > Align Classes to Middle command
	Align Classes to Bottom Tool	Aligns the bottoms of selected classes to the bottom-most class. Equivalent to the Arrange > Align Classes to Bottom command
	Align Classes to Left Tool	Aligns the left edges of selected classes to the left-most class. Equivalent to the Arrange > Align Classes to Left command
	Align Classes to Center Tool	Aligns the selected classes horizontally through the center of each class. Equivalent to the Arrange > Align Classes to Center command
	Align Classes to Right Tool	Aligns the right edges of selected classes to the right-most class. Equivalent to the Arrange > Align Classes to Right command

Icon	Command	Description
	Distribute Classes Horizontally Tool	Evenly spaces three or more selected classes along a horizontal axis. Equivalent to the Arrange > Distribute Classes Horizontally command
	Distribute Classes Vertically Tool	Evenly spaces three or more selected classes along a vertical axis. Equivalent to the Arrange > Distribute Classes Vertically command
	Distribute Connections Tool	Evenly spaces the endpoints of all associations and generalizations connected to selected classes. Equivalent to the Arrange > Distribute Connections command

► **To align business classes**

1. Select the appropriate number of classes (see descriptions in the table above).
2. Click an alignment icon.

Viewing Workshop Components

You can select the components of the business model you wish to view by using the commands on the View menu. By default, your workshop preferences specify which components are visible. You can either change the default settings (see [“Setting Workshop Preferences” on page 145](#)) or select the components you wish to view or hide from the View menu. A check mark next to the component indicates it will be displayed. To hide a component, select its menu command again.

Controlling Fonts

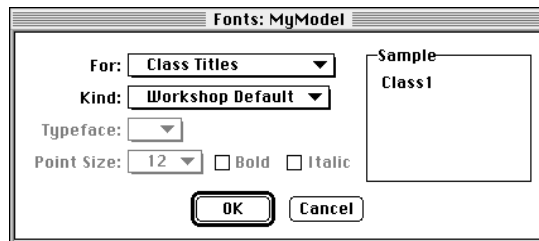
The File > Fonts... command lets you control the fonts used in the current business model. The fonts settings will override the default fonts set in the Workshop Preferences dialog. The File > Fonts... command displays a dialog with the following properties:

Property	Description
For	The For property specifies the area for which you are specifying the font. You can set the font for Class titles, attributes, and associations.
Kind	The Kind property specifies the type of font: Workshop Default (specified in the Workshop Preferences dialog), portable, or system.
Typeface	The Type property specifies the typeface.
Point Size	The Size property specifies the size of the font.
Bold/Italic	The Bold and Italic toggles specify whether the font is bold or italic.

► To set fonts for the current business model

1. Choose the File > Fonts... command.

The Fonts dialog appears.



2. From the For droplist, choose the area for which to set the font.
3. From the Kind droplist, choose the type of font to set.
4. Turn on the Bold and/or Italic toggles, if necessary.
5. Click OK.

Access to Other Workshops

By default, the TOOL classes that Express generates from your business model are read only. You can create a customizable class for an individual class or you can specify that Express automatically create the full hierarchy for every class. Once customizable business classes exist, you can enter the Class Workshop for a selected class with the Component > Class Workshop command.

For complete information about creating customizable Express classes, see *Customizing Express Applications*.

Leaving the Business Model Workshop

To exit the Business Model Workshop, use the File > Close command. You can also use the File > Cancel command, which closes the workshop and discards any changes you have made since the last save.

Business Classes

Business classes correspond to the database tables on which a business system is based. Express automatically generates TOOL classes based on the business classes. These TOOL classes define the data your application uses—each object of a given class contains a record from its associated database table.

If you want your application to have access to data in certain tables, then those tables must be represented by business classes in the business model. You can build one business model that represents your entire business system, or you can build several business models (application models can use business classes from several business models). See [Chapter 6, “Using the Application Model Workshop,”](#) for more information about using multiple business models.

Once you have created some or all of your business classes, you can use the Application Model Workshop (see [Chapter 6, “Using the Application Model Workshop”](#)) to build the client side of your application.

Creating a Business Class

You create a business class using the New Class tool or the New... slide-off menu under the Component menu.

➤ **To create a new business class**

1. Select the New Class tool and click in the window area.



You can also choose the Component > New > Class command.

Once you create a class, the class itself remains selected, the New Class tool is deselected, and the Select tool is activated. If you know you need to create multiple classes, and do not want to have to activate the New Class tool each time, follow the steps below:

➤ **To create multiple classes without reselecting the New Class tool**

1. Choose the Component > Repeat New command.

The Repeat New command remains selected for the workshop until you toggle the command off.

2. Select the New Class tool.
3. Click once in the window area for each class you need.

NOTE Remember, you can also use the sticky palette feature (*Ctrl-tool*) to create the same object repeatedly.

Business Class/Service List

The name of each business class you create or import also appears in Business Class/Service list. You can drag a business class from the Business Class/Service List into the Application Model Workshop (see [“Creating a Business Class Window” on page 172](#)). You can also use this list to locate a business class that does not appear on the screen—click on the class name and the model is repositioned to center the selected class in the window.

Also appearing in this list is the business service that Express creates by default when you create the business model. The service has the same name as the business model. For information about creating additional services and associating classes with them, see [“Express Services” on page 125](#).

Setting Class Properties

Every class has a name. By default, Express names the first business class you create Class1, the second Class2, and so on. If you import your business classes, Express gives the classes the same name as the tables on which they are based (see [“Importing and Exporting Database Schema” on page 135](#)).

Class Name property You can rename your business classes in the Class Name property of the Business Class Properties dialog. The name of the business class will be the basis for the names of the generated TOOL classes. You can also change the business class name using the Text tool.

Database Table property Every business class must be based on a database table. If you are basing your business model on an existing database, then each class must be based on a table in the current database. If you import your business classes, the Database Table property is set automatically. If you will be exporting your model to generate database schema, then Express creates a script in which a database table will be created and named after each business class.

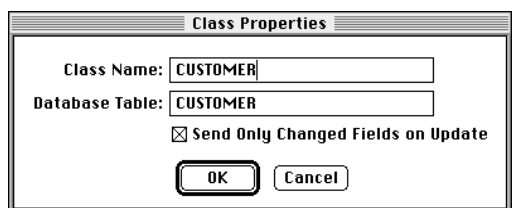
Send Only Changed Fields on Update property Express uses business classes and their attributes to build the SQL statements that select from or update the database. The Send Only Changed Fields on Update property tells Express to send only the information the database needs to perform the update. For example, if a record contains ten fields, but only three of them have been changed, Express sends only the data from the three changed fields to update the database. Using the property improves application performance because it reduces network traffic. By default, this property is set to on. However, for some advanced customizations, it may be useful to send all data to the server when updates are performed. You can turn off this property in these situations.

You can edit the business class properties in the Class Properties dialog.

➤ To set or modify class properties

1. Double-click on the class, or select the class and then choose the Component > Properties... command.

The Class Properties dialog appears, as shown below.



Attributes

Every business class must have at least one attribute. As explained in *A Guide to the iPlanet UDS Workshops*, an attribute is a data item. In a business class, the data in the attributes corresponds to the columns in the database table on which the business class is based. A business class can contain attributes for as many of the database columns you need for your application. You create attributes and set their properties in the Attribute Properties dialog (see [“Creating an Attribute” on page 115](#)).

By default, all attributes in a business class appear as fields on the generated window. See [Chapter 6, “Using the Application Model Workshop,”](#) for more information about attributes and their relationship to generated windows.

Database Column property The Database Column property in the Attribute Properties dialog specifies the name of the column to which the attribute corresponds. If you have imported a database schema, this name is by default the name of the column on which the attribute is based. If you are exporting a database schema, this name will become the column name in the database table.

Attributes in a business class correspond to attributes in generated TOOL classes. Express can map TOOL attributes to columns in a database table.

Primary Keys

Every business class, except a subclass in a generalization relationship, must have a primary key. Express uses the primary key to implement mapping between database tables and iPlanet UDS classes. Otherwise, there is no correspondence between database rows and business class objects.

Key property The Key property in the Attribute Properties dialog specifies which attribute is the primary key. Primary keys can be composite—simply assign the Key property to each attribute you want to include. If you do create a composite primary key, the order in which the attributes appear in the business class determines the order of the composite key. You can reorder a composite primary key by selecting an attribute on a business class and dragging it to a new position.



If you import database tables that include primary key definitions, those columns defined as primary keys will appear as business class attributes with a key icon to their left. If your tables do not include primary key definitions and you wish to create them, simply open the Attribute Properties dialog for the appropriate attribute and check the Key check box.

Custom attributes cannot be primary keys (see [“Custom Attributes” on page 115](#)).

Attribute Types

Just like regular class attributes, business class attributes have types. Since business class attributes usually appear as fields on a window, they use a special type called a *domain*.

Domain property A domain is a special class that combines a `DataValue` subclass with an iPlanet UDS widget. Express uses domains as the types for attributes, because attributes will always be used in conjunction with widgets for display on application windows. The following domain classes are provided by iPlanet UDS; you select one of the domains from the Domain drop list on the Attribute Properties dialog. To create your own domain classes, refer to *A Guide to the iPlanet UDS Workshops*. To assign a custom domain class to an attribute, see [“Assigning Custom Domains” on page 116](#).

Domain	Default Widget	Stores and Manipulates
<code>BooleanDomain</code>	<code>ToggleField</code>	Logical values of <code>TRUE</code> and <code>FALSE</code> and the <code>NULL</code> value.
<code>DateTimeDomain</code>	<code>DataField</code>	Date time data and the <code>NULL</code> value.
<code>DoubleDomain</code>	<code>DataField</code>	Floating point data and the <code>NULL</code> value.
<code>ImageDomain</code>	<code>PictureField</code>	Image data in a standard, portable format, and the <code>NULL</code> value.
<code>IntegerDomain</code>	<code>DataField</code>	Whole numbers of any size and the <code>NULL</code> value.
<code>LongTextDomain</code>	<code>TextField</code>	Long character data and the <code>NULL</code> value.
<code>MoneyDomain</code>	<code>DataField</code>	Money data and the <code>NULL</code> value.
<code>TextDomain</code>	<code>DataField</code>	Character data and the <code>NULL</code> value.

Field Label Text and Width

When you create a window class based on a business class, the attributes in the business class are available to appear as fields on the generated window.

Field Width property You use the Field Width property in the Attribute Properties dialog to specify the width of the field on the window. If the business class was imported from a database, then the width of the database column is the default Field Width.

Controlling Generated Field Width Domains whose widget is a `CharacterField` or a subclass of `CharacterField`—`DataField`, `FillInField`, `TextField`, and `TextEditField`—have widget properties relevant to controlling field widths on generated windows, namely:

- **Max Characters**—The maximum number of characters a user can type into a text field.
- **Visible Columns**—The number of columns of text visible in a text field at any given time.
- **Visible Lines (TextField widget only)**—The number of lines of text visible in a text field at any given time.

(See the Display Library online Help manual for a class description of these widgets.)

The settings of these properties combine with the **Field Width** setting (in the **Attribute Properties** dialog) to produce generated field widths as follows:

If you set **Field Width** to non-zero:

- The new setting becomes the **Max Characters** value for the generated field.
- For `DataField` and `FillInField` domains, the new setting also becomes the **Visible Columns** value for the generated field.

If you set **Field Width** to 0:

- The values specified in the domain are used.

Field Title property The **Field Title** property in the **Attribute Properties** dialog specifies the text of the label that appears to the left of a field on form windows. When you create an attribute, Express fills in the attribute name for this property. (Note that the text does not appear until you tab out of the **Attribute Name** field.) You can edit this property by deleting the default text and entering a new name. If you delete the default label and leave this property empty, no field label will appear on the generated window.

Column Heading property The **Column Heading** property in the **Attribute Properties** dialog specifies the text that appears above columns in array and outline windows. When you create an attribute, Express fills in the attribute name for this property. (Note that the text does not appear until you tab out of the **Attribute Name** field.) You can edit this property by deleting the default text and entering a new name.

Custom Attributes

A custom attribute is an attribute that does not have a corresponding database column, but rather is derived in the server code. For example, a custom attribute might be a figure calculated from other attributes or from data entered by an end user.

When you design your application windows in the Application Model Workshop, you use custom attributes as you would any other attribute. Before you run your application, you must open the Class Workshop to add the code that defines the data the attribute will contain. See *Customizing Express Applications* for information about creating the logic to support custom attributes.

Custom property To create a custom attribute, select the Custom toggle in the Attribute Properties dialog in addition to setting the standard attribute properties described above. The Database Column property will become unavailable when you select the Custom toggle. Note that custom attributes cannot be primary keys; therefore, the Key toggle will also become unavailable when you select the Custom toggle.

Creating an Attribute

When you create an attribute, the Attribute Properties dialog appears. Fill in the appropriate properties and click OK to create the attribute.

► To create an attribute

1. Select a class or an attribute and then click the New Attribute icon in the toolbar.

You can also choose the Component > New Attribute... command, or, if there are no attributes, double-click in the attribute area of the class. The Attribute Properties dialog appears, as shown below.

The screenshot shows the 'Attribute Properties' dialog box with the following fields and controls:

- Attribute Name:** A text input field followed by a checkbox labeled 'Key'.
- Domain:** A dropdown menu with a list icon to its right.
- Field Width:** A text input field containing the number '0' followed by the text 'characters'.
- Field Title:** A text input field.
- Column Heading:** A text input field.
- Database Column:** A text input field followed by a checkbox labeled 'Custom'.
- Buttons:** Three buttons at the bottom: 'OK', 'New', and 'Cancel'.

2. Enter the appropriate properties.
3. Click the OK button.

► **To create multiple attributes**

1. Follow steps 1 and 2 above.
2. Click the New button.

The New button clears the dialog and allows you to enter information for the next attribute.

3. After you enter the last attribute, click the OK button.

NOTE If you add attributes to a business class after you have created a business class window based on that business class, the attributes will not be displayed on the window by default. You must make the attributes visible in the Attribute List in the Application Model Workshop; see *“Specifying Window Fields in the Attribute List”* on page 180.

Assigning Custom Domains

You create a custom domain in the Class Workshop. See *A Guide to the iPlanet UDS Workshops* for information about creating domains. Once you create the domain class, you can assign it as an attribute type.

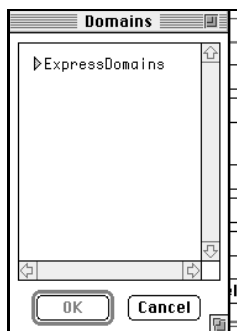
► **To assign a custom domain to an attribute**

1. Include the project that contains the domain class as a supplier to the business model.
2. Open the Attribute Properties dialog.

Type in the name of the domain, or follow steps 3-5 below.

3. Click the browser button next to the Domain attribute.

The Domains dialog appears.



4. Open the project containing the domain.
5. Highlight the domain type with which to associate the attribute and then click OK.

Editing Existing Attributes

At any time, you can edit an existing attribute. Simply activate the Select tool, double-click the attribute to bring up the Attribute Properties dialog, and make your changes. You can also select the attribute with the Select tool and then choose the Component > Properties.... command. If you just want to change an attribute name or domain, you can simply select the Text tool, click in the attribute name or domain and make your change.

Associations

An association describes the relationship between two classes. Express associations have the following properties: name, multiplicity, direction, and aggregation. Express supplies default values for these properties, and in many cases you will not need to make any changes.

Express also supports a special type of relationship called a generalization. See [“Creating a Generalization” on page 125](#) for information about generalizations.

From-class and to-class In the discussion of associations, the terms from-class and to-class are used. The to-class is the class at the end of the association at which the arrow appears. The from-class is the class at the non-arrow end of the association. The implications of the direction of the arrow are discussed in [“Direction” on page 118](#).

The default preference of the Business Model Workshop is to hide the arrows that represent the direction of the association. You can turn on these symbols either by changing the preferences setting (see [“Setting Workshop Preferences” on page 145](#)) or by using the View menu.

The following sections describe several association properties, which you set in the Association Properties dialog. See [“Creating an Association” on page 123](#).

Associations and Generated Classes

As mentioned in [“Overview” on page 33](#), and explained in detail in *Customizing Express Applications*, Express generates several TOOL classes based on the business classes and relationships between them. The direction of an association, as visually represented by the arrow, determines the details of how Express will implement the association in the generated TOOL code for the business class. The multiplicity also affects the generated code.

Direction

When Express generates TOOL code for the classes described in the business model, it creates attributes in the generated classes. These attributes are based on associations. For example, in [Figure 5-6 on page 120](#), there is an association (called Customer) from CustomerOrder to Customer. Thus, the generated class CustomerOrderClass will have an attribute called “Customer” that points to a CustomerClass object. Based on the association in [Figure 5-7 on page 120](#), the CustomerOrderClass will have an attribute called LineItem that points to many LineItem objects (an array). In general, the from-class will always contain an attribute that references the to-class.

The class that contains the attribute corresponding to the association may be important to your application logic and behavior. For example, a lookup window from CustomerOrder to Customer will use the Customer attribute on the CustomerOrder class to perform the search for a customer associated with an existing order. On the other hand, you can construct a master-detail window that shows all the orders for a customer, regardless of the direction of the association between CustomerOrder and Customer.

Another affect of association direction is that in the Application Model Workshop, the attributes of the to-class will be available as read-only fields on a window based on the from-class. See [“Specifying Window Fields from Two Business Classes” on page 183.](#)

Association Name property By default, the association is named after the to-class. In generated code, the association name becomes the name of the attribute on the generated from-class that references objects of the to-class. You can change the association name by editing the Association Name property in the Association Properties dialog.

Direction property The Direction property allows you to override the default direction of the association. Express assigns a default direction based on the primary and foreign keys of the business classes.

From Attribute and To Attribute property The From Attribute and To Attribute array field specifies the attributes on which the association is based. These attributes are used to construct a SQL join to retrieve the associated data from the database tables that correspond to their business classes. In a one-to-one or many-to-one relationship, the join must be based on the primary key of the “to-class.” If the “to-class” has a composite primary key, then all the key attributes must be used for the join, in the same order they appear on the business class.

Multiplicity

Multiplicity defines how many instances of one object can be associated with an instance (or instances) of another object. Relationships between business classes are referred to as *one to one*, *one to many*, and *many to one*. There is also an *optional* relationship, which is used to perform outer joins.

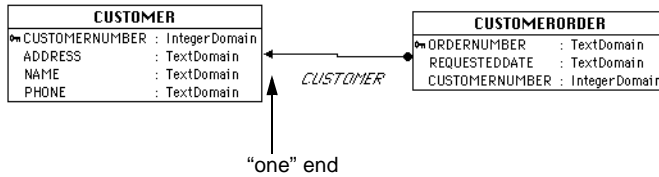
The multiplicity of the to-class determines the type of attribute that will be generated in the corresponding TOOL class.

To-One Associations

A *to-one* association is any association where the to-class has a multiplicity of one (one-to-one, many-to-one). Express assumes there is one record in the database that will satisfy the join criteria; this is referred to as an equi-join.

For example, imagine an order entry system that contains a Customer business class and a CustomerOrder business class. Each customer can place many orders—this relationship is one-to-many. The Business Model Workshop uses a dot (•) to indicate the “many” end of the relationship. If there is no dot, then the multiplicity of that class is one. The figure below illustrates:

Figure 5-6 Example of Multiplicity—Many to One



The CustomerOrder TOOL class will have an attribute that is a reference to a single Customer object, but, many orders can be placed by the same customer (as is indicated by the dot on the from-class).

If the multiplicity was one-to-one, then there should be only one CustomerOrder object for each customer; however, Express does not enforce this in the Business Model Workshop.

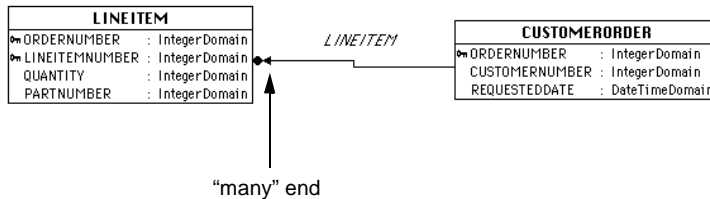
Another characteristic of one-to-one or many-to-one multiplicity is that the attributes of the to-class can be displayed on the generated window of the from-class. See [“Specifying Window Fields from Two Business Classes” on page 183](#). In addition, the multiplicity will determine which types of window links will be available in the application model. See [“Associations and Links” on page 189](#).

To-Many Associations

A *to-many* association is any association where the to-class has a multiplicity of many.

Look at the association between CustomerOrder and LineItems in [Figure 5-7](#). The one-to-many association means that one order can have many line items. The to-class (LineItem) has a multiplicity of many (as is indicated by the dot), thus, there will be an attribute on each CustomerOrder object that references an array of LineItems.

Figure 5-7 Example of Multiplicity—One to Many

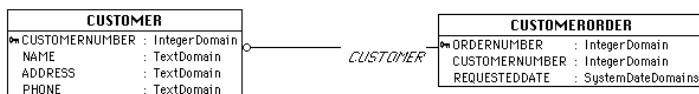


Multiplicity property You select the multiplicity of an association from the Multiplicity drop list in the Association Properties dialog. Express assigns a default multiplicity to an association based on the primary keys of the business classes. For an explanation of how the defaults are assigned, see [“Default Direction and Multiplicity” on page 122](#).

Optional Associations

An optional relationship represents either a *one-to-zero* or *one-to-one* relationship. That is, a business object may or may not have an associated record. You use an optional relationship when you want to ensure that your application returns records, even if no associated records exist; this is referred to as an outer join.

Figure 5-8 Example of Optional Association



Aggregation

Another characteristic of an association is aggregation. An aggregate association describes a relationship between two classes in which one class (the aggregate class) contains as sub-components one or more instances of the other class (the component class). Instances of the component class cannot exist without an instance of the aggregate class.

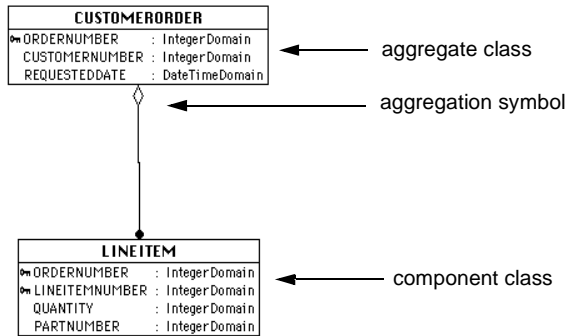
For example, in the order entry system, an order contains line items. The CustomerOrder class is the aggregate class, and LineItems is the component class—there can be no line items if there is no order.

NOTE If you delete a record of the aggregate class, Express does not automatically delete corresponding records in the aggregate's component classes. If you want these records to be deleted automatically, you must implement cascading deletes through local or global customization of the generated Express application code, or by implementing rules in your database that cause cascading deletes. In some databases, deletes of the aggregate components can be implemented as a database integrity rule.

Aggregation property You specify that an association is an aggregation by selecting the Aggregate toggle.

An aggregation is indicated by a diamond symbol at the aggregate end of the association, as shown below.

Figure 5-9 Aggregation

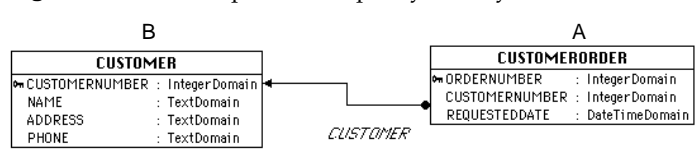


Default Direction and Multiplicity

Express assigns a default direction and multiplicity for an association based on the primary keys and other attributes it detects on classes either imported from the database or created by the user. In general, Express attempts to match attributes by name. The following examples explain the rules used for assigning a default direction.

Many to one If a non-key attribute name in one class (class A) matches the primary key of another class (class B), then this is a many to one association, with the arrow pointing toward class B (which is at the “one” end), as shown in [Figure 5-11](#). This is often characterized as a “lookup” relationship. See [“Link Types” on page 186](#).

Figure 5-10 Example of Multiplicity—Many to One



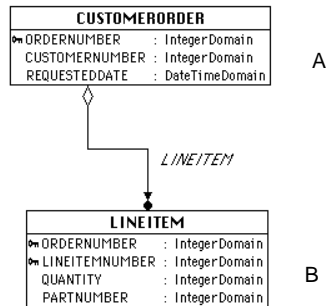
One to one If a complete primary key in class A matches the complete primary key in class B, then this is a one to one relationship.

One-to-many aggregate Consider the following:

- the primary key attributes in one class (class A) match the first primary key attributes in a composite primary key in another class (class B)
- the number of keys in A is less than the number of keys in B
- the subset of keys in B is the same order as class A

If the above properties are true, then this is a one-to-many aggregate relationship. Class A is the composite class and the “from-class,” meaning the aggregate class is Class A, and the arrow points to class B.

Figure 5-11 One-to-Many Aggregation



All other scenarios are default to one-to-one relationships in the direction the association was drawn. In these situations, you will probably have to explicitly set some properties in the Attribute Properties dialog, as described in [“Setting Association Properties”](#) on page 124.

Creating an Association

You create an association using the New Association tool.

► To draw an association between two classes

1. Select the New Association tool.



2. Place the cursor over the first business class and hold down the mouse button.

3. Drag the cursor over the second business class and release the mouse button.

By default, the association has the same name as the class to which the arrows points.

Once you create an association, the association itself remains selected, its tool is deselected, and the Select tool is activated. If you know you need to create multiple associations, and do not want to have to activate the New Association tool each time, follow the steps below:

➤ **To create multiple associations without reselecting the New Association tool**

1. Select the Component > Repeat New command.

The Repeat New command stays active until you toggle it off, meaning that you can create multiple versions of any component.

2. Select the New Association tool.
3. Draw each association.

Once you create an association, you can modify its default properties.

Setting Association Properties

You set association properties in the Association Properties dialog.

➤ **To set association properties**

1. Double-click on the association.

The following dialog appears:

From Attribute	To Attribute
CUSTOMERNUMBER	CUSTOMERNUMBER

2. Enter or modify the appropriate properties and click OK.

Creating a Generalization

A generalization is a special type of relationship that defines a subclass. You use a generalization when you create a superclass and one or more subclasses. The subclasses inherit all the attributes defined on the superclass.

Express expects that the subclass (or subclasses) and the superclass are all based on the same database table. Therefore, if you open the Class Properties dialog for a subclass, there will be no Database Table property.

Note that when you draw a generalization, you start from the subclass and end at the superclass.

► To create a generalization

1. Draw or import a class that will be the superclass.
2. Create one or more classes that will be subclasses.
3. Select the New Generalization tool.

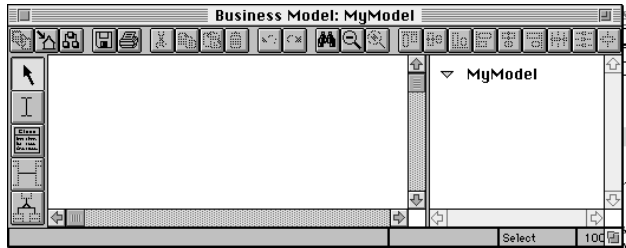


4. Drag the cursor from the subclass to the superclass.
5. Repeat for each subclass.

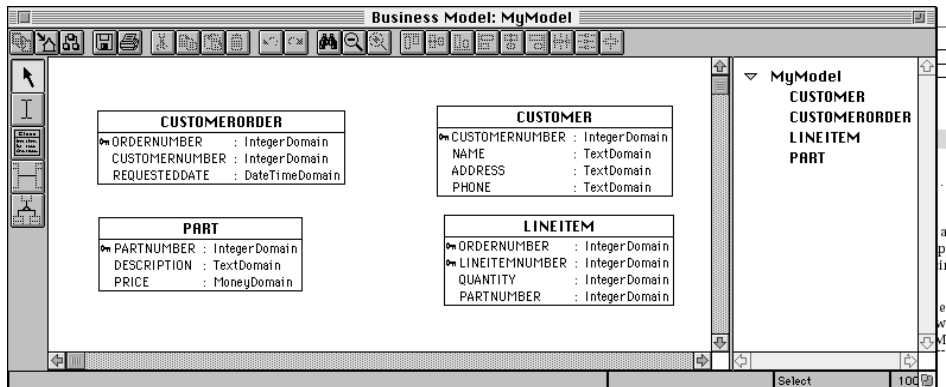
Express Services

A Express service is a central point of access and control for a set of related business classes. Express services are intended to be used as shared application services for a set of applications. An iPlanet UDS service object capable of load balancing and failover will be generated for each Express service.

Each time you create a new business model, Express creates a default service and displays it in the Business Class/Service list. For example, as shown in [Figure 5-12](#) below, a business model named MyModel will have a service called MyModel:

Figure 5-12 MyModel Default Service

By default, all business classes are managed by this service. If you import or create business classes, they will appear below in the browser, as shown below:

Figure 5-13 Classes Managed by MyModel Service

Service objects for Express applications When you generate your application, iPlanet UDS creates two service objects—a TOOL class service object and an associated DBSession service object—for each Express service in the application. The properties you set for the Express service using the Business Model Workshop affect the definitions of the corresponding service objects. See [Chapter 7, "Partitioning and Deployment,"](#) for information about working with the service objects in your Express application.

Service object properties are divided into three categories:

- general properties
- database properties
- connection properties

The properties for each category are described in the following sections.

General Properties

Service Name property Every Express service has a name, which, as mentioned above, is by default the same name as the business model. You can change this name by modifying the Service Name property in the Service Properties dialog.

Concurrency property A Service also controls the concurrency policy in effect for each class it manages. You control the concurrency setting with the Concurrency property in the Service Properties dialog.

Concurrency is the mechanism through which Express ensures that users have a consistent view of their data when multiple users access the same data. Express offers the following concurrency policies:

Optimistic: Verify This option is the default. When an update is performed, Express verifies that the data in the database has not changed since it was selected. If the data is unchanged, the update is successful. If the data has been changed, Express aborts the transaction and returns an exception stating that the row has been changed. Transactions are run only on the server, and no distributed transactions are involved. Message dialog duration and environment visibility are set automatically when you select this concurrency policy. These settings are the most scalable options for a large number of users.

Details on how Express determines whether data has changed are as follows:

- Additional qualifications are added to the update's where clause, reselecting the data and comparing the results to the columns originally selected.
- Not all attribute types are verified, for example, attributes of type LongTextDomain, ImageData, or BinaryData are not verified.
- Attributes of type DoubleData are checked for equality within 10^{-12} (that is, 0.000000000001). If your database column does not represent data to that accuracy, values that have not changed may appear to have changed when you try to save.
- For attributes of type TextData, trailing blanks are not significant.

Database: Native Locking This setting implements the concurrency policy in effect for the database your application is using. Express starts a transaction when records are selected and ends the transaction when the records are committed by a Save operation. Depending on the database and the isolation level in effect for that database, records may or may not be locked after a select. Depending on how the

database is configured, if multiple users select the same records, they may be able to save updates to them. However, if the save operation was successful, the second and succeeding users may be making updates based on stale data. Transactions begin and end on the client and spread to the server.

Database: Explicit Locking Express starts a transaction when records are selected and ends the transaction when records are committed by a Save operation. Every selected record is locked until a save is performed. Another user attempting to select a record that has not been saved will block waiting for the first user. No saves will be performed on stale data. This policy maintains integrity, but allows little concurrent access. Transactions begin and end on the client, and spread to the server.

None No concurrency policy in effect. This means that Express encloses every SQL statement in its own transaction. Updates to the database will be allowed in any order, regardless of how the data might have changed since the corresponding select.

Changing the concurrency policy sets the visibility and dialog duration to the correct settings for the selected policy. Do not change these settings unless you are customizing the code or configuration of your generated application.

Visibility The Visibility property specifies the visibility for the Express service, environment visibility or user visibility. See *A Guide to the iPlanet UDS Workshops* for background information on visibility of service objects.

The default visibility for the Express service is "Environment." To create a private partition for the Express service, you can change the Visibility property to "User."

Dialog Duration The Dialog Duration property specifies the dialog duration used for the Express service. The default dialog duration for the Express service as well as the values that are allowed for the particular service depend on the current value of the Service's Concurrency property. The following table shows the default dialog duration and the values allowed for the dialog duration for each of the concurrency modes:

Concurrency Mode	Default Dialog Duration	Values Allowed for Dialog Duration
None	Message	Message, Transaction, Session
Database: Native Locking	Transaction	Transaction, Session
Database: Explicit Locking	Transaction	Transaction, Session
Optimistic: Verify	Message	Message, Transaction, Session

Database Properties

Database Manager property The Database Manager property specifies the resource manager to use for the service. In the Service Properties dialog, the drop list for the Database Manager field displays all the external manager names that were previously defined in your environment. You can override this setting for the DBSession service object when you partition the application for a particular deployment environment in the Partition Workshop.

Database Name, User Name, Password properties The Database Name, User Name, and Password properties for the Express Service specify the database name, user name, and password for the database session used for the service. You can set these properties in the Service Properties dialog. You can override the database session properties setting when you partition the application for a particular deployment environment in the Partition Workshop.

Connection Properties

Option Name The Option Name property specifies an option that is processed when the connection to the database is established.

Value The Value property specifies the value of the option.

See [“Optional Environment Variables” on page 132](#) for information about setting connection options.

Creating New Express Services

By default, your Express application contains only one Express service that manages all business classes in the application. However, you may wish to create additional Express services, which can each manage a subset of the business classes. Common reasons for creating additional Express services are:

- Since each service is bound to a particular database session, you may wish to create multiple services, each of which are bound to different database sessions or even different database resource managers.
- Creating more than one Express service allows you to provide different replication options (load balancing or failover) for their corresponding service objects.

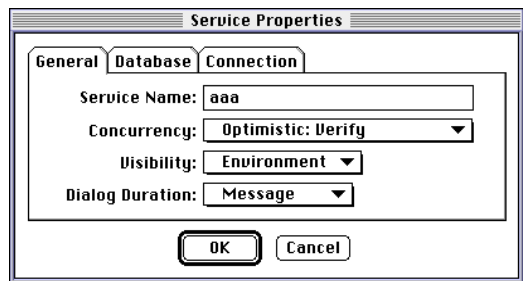
- Creating more than one Express service allows you to distribute processing across different nodes in the deployment environment by partitioning their corresponding service objects.

See [Chapter 7, “Partitioning and Deployment,”](#) for information about Express services in Express applications.

► **To create a service**

1. Choose the Components > New Service command.

The Service Properties dialog appears, as shown below:



2. Type the name of the service—this property is required.
3. Enter other appropriate service properties.

All properties other than Service Name are not required at the time you create the service, but are required before you generate the server classes.

4. Click OK.

The new service appears in the Business Class/Service list.

You are now ready to assign classes to the service, as described in [“Assigning Classes to a Service” on page 133](#), or you can create other services.

➤ **To set database properties**

1. Double-click on the new service in the service list to bring up the Service Properties dialog.
2. Click the Database tab.

The Database page appears.

Service Properties

General Database Connection

Database Manager: testOracle

Database Name: @HILLARY_ORACLE

User Name: system

User Password: *****

OK Cancel

3. Enter the appropriate properties and click OK.

➤ **To specify a connection option**

1. Click the Connection tab.

The Connection page appears.

Service Properties

General Database Connection

Option Name	Value

New... Delete

OK Cancel

2. Click the New... button.

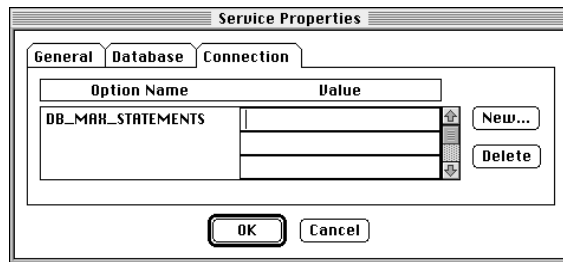
The New Connection Option dialog appears.

New Connection Option

Connection Option Name:

OK Cancel

3. Type a connection option name, or choose one from the drop list, and click OK.



4. Type a value in the Value field and click OK.

For a list of available options, refer to the ConnectDB method on the DBResourceMgr class in the *Accessing Databases* manual.

If you wish to enter multiple options, click the New... button again and repeat steps 3 and 4.

Optional Environment Variables

If you know that you will be creating many business models based on the same resource manager, database, user name, and password, you can specify environment variables for each of these properties. These environment variables will populate the Service Properties dialog each time you create a new business model; the only property you will enter is the business model name. You can override the environment variables by entering different values in the Service Properties dialog.

The environment variables are the following:

This Environment Variable	Sets This Property
FORTE_EX_DBRESOURCE	Database Manager
FORTE_EX_DATABASE	Database Name
FORTE_EX_DBUSER	User Name
FORTE_EX_DBPWD	User Password

Assigning Classes to a Service

By default, all classes are managed by the default Express service. If you create additional services, you can assign classes to them by simply dragging a class from the Business Class/Service list under the appropriate service, as shown in the figures below.

Figure 5-14 All Classes Assigned to One Service

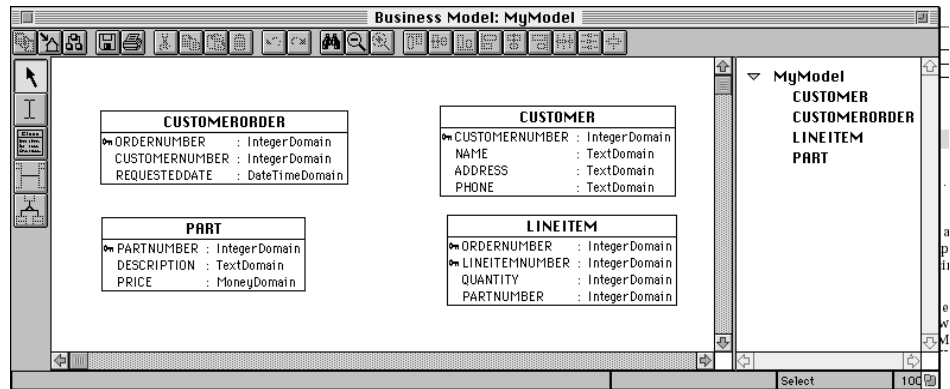
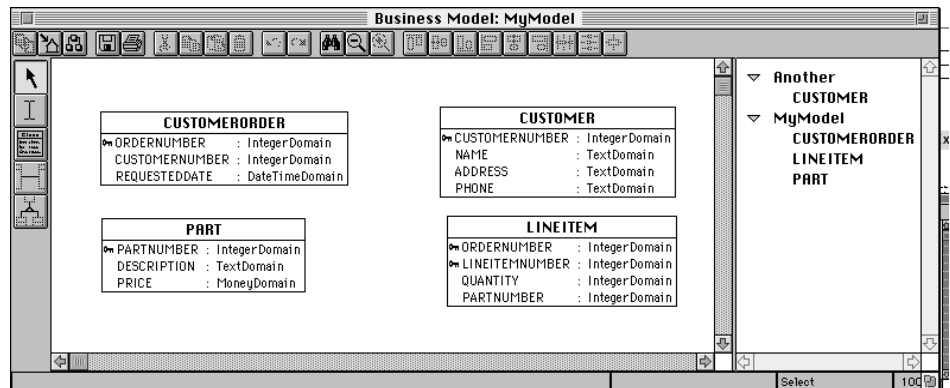


Figure 5-15 Assigning a Class to a New Service



Rules for Assigning Classes to Services

If you have an aggregate relationship in your business model, you must keep both classes in the same service. In addition, if you have a generalization in your business model, all the classes in the hierarchy must be managed by the same service.

Aggregate and generalized classes For example, to reassign aggregate or generalized classes to a service other than the default, you must drag the parent class to the appropriate service. When you move the superclass or composite class, the subclasses or aggregate classes will move with it. When classes are in the same service, you can write code to define transactions that involve multiple classes in the same service. This is important for some types of server-side customizations.

Editing Service Properties

You can edit the properties of a service at any time by double-clicking on the service, or selecting the service and choosing the Properties... command from the component menu.

Deleting Services

Before you delete a service, you must move all the classes it manages to another service. Every business model is required to have at least one service. You cannot delete the last service from a business model, even if it does not manage any classes.

- **To delete an empty service**
 1. Select the service you wish to delete.
 2. Click the Delete icon.

Importing and Exporting Database Schema

Express provides two commands to facilitate working with a database: Import Database Schema and Export Database Schema.

Importing Database Schema

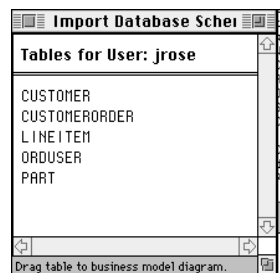
If you have an existing database, you can import the database table definitions into the Business Model Workshop. iPlanet UDS converts the table definitions into business classes, and the table column definitions into attributes. Importing the database schema allows you to quickly build the foundation of your business model. When you import database schema, iPlanet UDS provides default class property settings, which you can use as is or modify.

NOTE If you are importing tables from a Sybase database, the account your model is using must have connect permission to the system catalogs. See your System Administrator for information.

► To import database schema

1. Choose the File > Import Database Schema command.

The Import Database Schema window appears, as shown below.



2. Select a table you wish to use in the model, then drag and drop its name into the Business Model Workshop.
3. Repeat step 2 for each table you wish to include in your model.
4. Close the Import Database Schema dialog when you finish importing the tables.

iPlanet UDS creates a business class with the same name as the database table. If the underlying database does not support case-sensitive names, then the names are displayed in upper case. Each column in the table is represented as an attribute on the business class.

Primary keys If the database table contained a primary key, Express captures that information and displays a key icon next to the attribute that represents the primary key. If a table does not have a primary key, you must create one using the Attribute Properties dialog (see [“Primary Keys” on page 112](#)).

Importing Schema from Multiple Databases

If you plan to import database schema from more than one database, you must create a service for each database—*before* you import the schema. For information about creating new services, see [“Express Services” on page 125](#). When you import the schema for each database, the classes will be managed by the service you select.

► To import schema from multiple databases

1. Create a new service.
2. Select the service and choose the File > Import Database Schema command.

(If you have not defined the service properties, you are prompted to do so now.)

The Import Database Schema dialog appears.

3. Drag the appropriate tables into the business model, as described in steps 2-4 above.

Exporting Database Schema

While you often base your business model on an existing set of database tables, you can also create a model unrelated to database tables, and then create new tables based on the business model. Express generates a script of create table statements that reflect the definition of the business classes that comprise the business model. This script can be run with any command line interface (such as Oracle’s SQL*Plus or Sybase’s Isql) to create database schema.

Relationship Between Business Classes and Exported Database Schema

For each business class, one create table statement is created. The table name is the same as the name you specify in the Business Model Properties dialog. Each attribute in a business class corresponds to a database table column; columns are created in the same order as the attributes listed in the business class.

The primary key statement is placed at the end of create table statements; the clause includes all columns marked as Key in the business class, in the same order they appeared in the class.

If your model uses generalizations, one create table statement is created using the entire hierarchy, with a composite of all the columns. All columns that correspond to attributes in the superclass appear first, followed by the columns corresponding to the attributes from the subclasses. Note that primary keys can be assigned only to the superclass; thus, the primary key statement appears at the end of the create table statement for each attribute in the superclass marked as Key.

Associations are not represented in the database schema.

Figure 5-16 is an example of a script that is created when you export the Tutorial business model.

Figure 5-16 Tutorial Schema

```

mySchema
--
-- Schema = Tutorial, Database = hillary.FORTE, Vendor = Oracle
--

create table CUSTOMER (
  CUSTOMERNUMBER number(6,0),
  NAME          varchar(25),
  ADDRESS       varchar(25),
  PHONE         varchar(12),
  PRIMARY KEY (
    CUSTOMERNUMBER
  )
);

create table CUSTOMERORDER (
  ORDERNUMBER number(6,0),
  CUSTOMERNUMBER number(6,0),
  REQUESTEDDATE date,
  PRIMARY KEY (
    ORDERNUMBER
  )
);

create table LINEITEM (
  ORDERNUMBER number(6,0),
  LINEITEMNUMBER number(3,0),
  QUANTITY number(13,0),
  PARTNUMBER number(6,0),
  PRIMARY KEY (
    ORDERNUMBER,
    LINEITEMNUMBER
  )
);

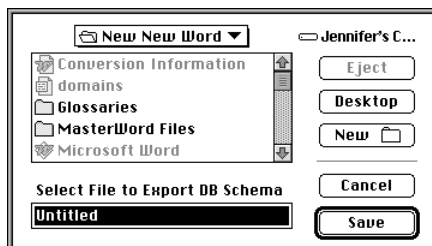
create table OLDPART (
  PARTNUMBER number(6,0),
  DESCRIPTION varchar(25),
  PRICE decimal(11,0)
);

```

► **To export database schema definitions**

1. In the Business Model Workshop, choose the File > Export Database Schema command.

A standard file selection dialog appears, as shown below.



2. Type the name of the file in which to store the script.
3. Click the Save button.

Exporting Schema from Multiple Services

As mentioned in *“Express Services” on page 125*, a business model has one service by default; you can add services if necessary. If you are using multiple services, you have two options for exporting database schema. First, you can simply select the Export Database Schema command and the export will work as described above. All the classes in the model will have corresponding create table statements in the script. The second option is to export only the classes managed by a specific service. To do this, first select the service whose classes you wish to export, and then choose the Export Database Schema command. A dialog appears prompting you to either export the business classes associated with all the services (in other words, all the classes) or just the classes associated with the service you have selected at the time of the export. Choose the appropriate option.

Server Code Generation

Express automatically generates TOOL classes from a business model. Prior to generating your code, you can validate the model to check its consistency. If you have customized classes in ExpressServices, you need to set a custom prefix prior to code generation. You can also force a complete generation to regenerate imported TOOL classes. Each of these topics is described below.

Validating the Model Prior to Generation

The Validate Model command allows you to verify the completeness and consistency of the business model before generating the server code.

For the business model to be valid, it must meet certain requirements:

- all classes must have primary keys and must be associated with different tables
- all attributes must have column names
- all attributes referenced by associations with to-one multiplicity must be primary keys
- every service must have a valid DB resource name, database name, user name, and password within the current environment

If there are any errors in your model, they will be displayed in the Errors Found During Model Validation window. You must fix these errors before you can give a Generate Server Code or Force Complete Generation command.

Note that the Generate Client Code and Force Complete Generation commands automatically execute the Validate Model command. If the model is invalid, you must fix the errors before these commands can generate the code.

► To validate the business model

1. Select the Validate Model command from the File menu.
2. If there are any errors, these are displayed in the Errors Found During Model Validation window.

Potential Error Messages for Business Model Validation

Error Message	Explanation
Class <i>name</i> must have primary key.	The specified class does not have a primary key. For the appropriate attribute in the class, open the Attribute Properties dialog and turn on the Key toggle.
Classes must have different tables. Class <i>name</i> and class <i>name</i> both have table <i>name</i> .	The specified class are both associated with the same table. For the appropriate class, open the Class Properties dialog and change the database table name.
Attribute <i>name</i> must have a column name.	The specified attribute is not associated with a column name. Open the Attribute Properties dialog, and enter the appropriate column name.
All attributes referenced by associations with to-one multiplicity must be all primary keys of the To class. Association <i>name</i> references attribute name which is not a primary key of class name.	The attribute being referenced by the association is not a primary key of the specified class. Therefore, you must change the association to reference the class's primary key. Open the Association Properties dialog and change the To class to the primary key.
All attributes referenced by associations with to-one multiplicity must be all primary keys of To class. Association <i>name</i> does not reference primary key <i>name</i> of class <i>name</i> .	The association is not referencing the <i>entire</i> composite primary key of the specified class. You must open the Association Properties dialog, and set the To class to the complete composite primary key.
A valid value for the Database Manager in the Service is required.	Blank values are not allowed for the database resource name for a service. You must open the Service Properties dialog and enter a valid value for the Database Manager field.
A valid value for the Database Name in the service is required.	Blank values are not allowed for the database name for a service. You must open the Service Properties dialog and enter a valid value for the Database Name field.
A valid value for the User Name in the service is required.	Blank values are not allowed for the user name for a service. You must open the Service Properties dialog and enter a valid value for the User Name field.

Setting a Custom Prefix for Generated Classes

As described in *Customizing Express Applications*, you can modify the following classes from the ExpressServices project to alter the behavior of your generated classes:

- BusinessClass
- BusinessClient
- BusinessDBMgr
- BusinessQuery
- BusinessServiceMgr

ExpressServices is a supplier plan to every business model.

To customize these generated classes, you first create a new project that contains subclasses of the classes mentioned above. You must name the subclasses *yourPrefixClass*, *yourPrefixClient*, *yourPrefixDBMgr*, *yourPrefixQuery*, and *yourPrefixServiceMgr*. Note that all the classes must have the same prefix. Make your modifications to the subclasses and include the project as a supplier to your business model.

Before you generate code, you must use the File > Custom Generation Options command and enter the prefix (*yourPrefix*) in the Custom Generation Options dialog.

► To enter a custom generation prefix

1. Choose the File > Custom Generation Options command.

The Custom Generation Options dialog appears. If you have not previously changed the prefix, the prefix will be “Business” by default.



2. Enter the same prefix you used in the name of your custom classes.



3. Click OK.

Express searches the supplier projects and looks for classes with the prefix you specified in the Custom Generation Options dialog. All generated business classes will be subclasses of *yourPrefixClass*, *yourPrefixClient*, *yourPrefixDBMgr*, *yourPrefixQuery*, and *yourPrefixServiceMgr*.

Generating Server Code

After you have created your business model, you can generate the iPlanet UDS classes derived from the business classes.

For each business class, Express generates three read-only classes. Using the CUSTOMERORDER business class as an example, the generated server classes are:

Generated Class	Superclass	Description
CUSTOMERORDERClass	BusinessClass	Stores the data. Each CUSTOMERORDER object stores a record from the associated table.
CUSTOMERORDERQuery	BusinessQuery	Describes the database queries.
CUSTOMERORDERMgr	BusinessDBMgr	Manages database operations. This class is referred to as the Business Manager. Business managers manage select and update operations on the database. Business managers retrieve data from the database into business objects.

Customizable subclasses To customize any of these classes, you must create customizable subclasses of these classes. You can do this with one of the following methods:

- Invoking the Customization Manager on an individual class.

This renames the generated class, inserting “Base” into the name, and creates a new, customizable class, which is named with the original class’s name. For example, the CUSTOMERORDERClass (read-only) becomes CUSTOMERORDERBaseClass; the new customizable, leaf-level class is then named CUSTOMERORDERClass.

- Create a full hierarchy (Base and leaf-level classes) for every class.

You do this by setting the Always Generate Custom Classes option on the File > Custom Generation Options command.

When you use this option, note:

- Turning it on turns it on for all the business classes in the model; it can then only be turned off by deleting all the customizations on each business class using the Customization Manager. If you delete the individual classes you will get errors.
- In other words, you create customizable classes for all business classes in the model with one step, but you must remove individual customizable business classes separately.
- Turning it off will only affect new business classes created from the time you turned it off, resulting in some Base classes having “Base” in their names and some not.

For complete information on creating customizable classes and using the Customization Manager, see *Customizing Express Applications*.

➤ To generate server code

1. Choose the File > Generate Server Code command.

Express creates the *business_modelServices* project, which contains all the iPlanet UDS classes generated from the business model. Refer to *Customizing Express Applications* for information about the generated classes.

Note that you are not required to generate the server code before you build your application model. In fact, you can generate the client code in the Application Model Workshop, and Express will automatically generate the server code along with it.

Forcing Complete Code Generation

If you want to guarantee that all classes in a model will be regenerated, choose the File > Force Complete Generation command. After you import TOOL projects that contain generated classes into your repository, you should use the Force Complete Generation command to regenerate the entire model. The imported TOOL projects may contain previously generated code that is not up to date with the current version of Express. Forcing complete generation will bring the imported project back up to date.

Setting Workshop Preferences

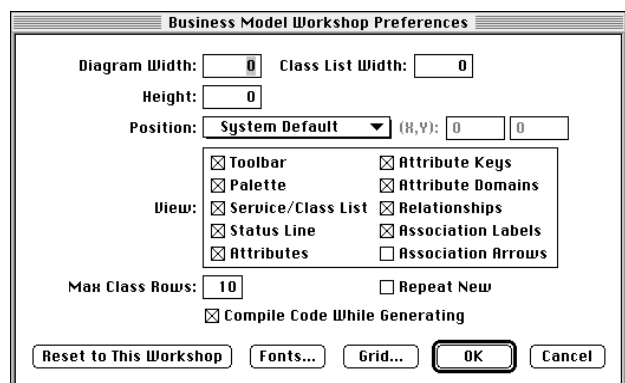
You can set certain visual and behavioral aspects of the Business Model Workshop to suit your development style. Once you set your preferences, the Business Model Workshop will reflect them each time you open the workshop.

You set Business Model Workshop preferences in the Business Model Workshop Preferences dialog.

► To set Business Model Workshop Preferences

1. Choose the File > Workshop Preferences... command.

The Business Model Workshop Preference dialog appears.



Each preference option is briefly described in the following table, and explained in more detail in the sections that follow.

Preference	Description
Diagram Width	Sets the default width of the drawing area.
Height	Sets the default height of the drawing area.
Class List Width	Sets the default width of the class list.
Position	Sets the initial position of the Business Model Workshop window in relation to other windows or the screen.
X,Y	Sets the precise positioning a window should use to place itself, according to the Position setting.
Repeat New	Toggles the Repeat New command in the Component menu to on by default.
View	Specifies which diagram components to view by default.
Max Class Rows	Specifies the number of attributes to display in a business class before adding scroll bars to the business class.
Compile Code While Generating	Specifies whether or not to compile while generating either server or client code. The default is on. If you do not select this property, Express compiles the code at runtime.
Reset to This Workshop	Sets all preferences to that of the current workshop.
Fonts	Sets the fonts used in the Business Model Workshop.

Workshop Size and Position

There are several different ways you can set the initial size and position of the Business Model Workshop: capturing the settings of the current workshop, explicitly setting the dimensions of the drawing area and the class list, and specifying the X and Y coordinates of the workshop relative to a specified position.

Capturing Current Workshop Settings

The simplest way to set the initial size and position of the workshop is to do so visually. You can set the window size by using the window's resize handles, and the window position by dragging the window into position. When the window reflects the size and position you desire, choose the File > Business Model

Workshop Preferences command and click the Reset to This Workshop button in the dialog. This button sets the values of the Position preference to Screen Relative and the values of the X, Y coordinates to the current values of the workshop on the screen.

Note that the Reset to This Workshop button changes all preferences to the current settings in the Business Model Workshop.

Setting Dimensions of the Drawing Area

You can control the size of the workshop by specifying the size of the drawing area and the class list area:

Preference	How to specify its value
Diagram Width	Enter an integer that specifies the width of the Business Model Workshop drawing area in mils.
Height	Enter an integer that specifies the weight of the Business Model Workshop drawing area in mils.
Class List Width	Enter an integer that specifies the width of the Class List in mils.

Specifying Workshop Position

By setting the Position property, you determine the initial placement of the Business Model Workshop window when the window first runs. You set the window's initial position either in relation to the screen or an application's primary window, or absolutely.

Relative initial window position To set the Business Model Workshop window's position relative to either the screen or to the primary window, you use the Position setting in conjunction with the X and Y specifications. The X setting determines (in *mils*—thousandths of an inch) how far to offset the window horizontally, relative to the screen or a primary window. The Y setting determines the vertical offset. The primary window is normally the window that opened the current window—in this case it is the Repository Workshop.

Absolute window position To set a window's position absolutely, you choose a position setting of Screen Centered or Primary Centered. These absolute settings ignore the X and Y settings, if any.

The following table summarizes the window position settings.

Position Setting	Definition
System Default	The default position used by the particular window system on which the window is being displayed.
Screen Centered	Centers the window relative to the screen.
Primary Centered	Centers the window relative to the primary window.
Screen Relative	Positions the window relative to the screen, using the X and Y settings to offset the window from the screen.
Primary Relative	Positions the window relative to the primary window, using the X and Y settings to offset the window from the primary window.

Viewing Preferences

You can specify the default settings for menu items on the View menu in the Business Model Workshop. You can make each of the following components visible or invisible:

- Attributes
- Attribute Domains
- Attribute Keys
- Class List
- Palette
- Relationships
- Association Labels
- Association Arrows
- Status Line
- Toolbar

Click the checkbox of the components you wish to view; click again to remove from view. You can change this selection on the fly using the View menu.

Drawing Preferences

The Repeat New command in the Component menu allows you to create multiple copies of a business model component without having to reselect its drawing tool. To turn this command on by default, select the Repeat New toggle in the Business Model Preferences dialog.

Max Class Rows The Max Class Rows property specifies the number of attributes to display in a business class before adding scroll bars to the business class.

Compiling Preferences

The Compile Code While Generating preference tells Express to compile the code as it is being generated. By default, this preference is on. If you do not use this preference, Express compiles the code at runtime.

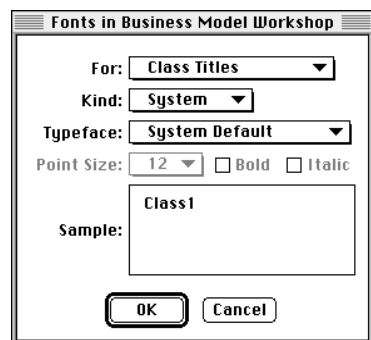
Font Preference

You can set the font to be used for components in a business model. You can individually set font characteristics for class labels, attribute names, association labels, and the class list.

► To set the font for the Business Model Workshop

1. Choose the File > Workshop Preferences... command.
2. Click the Fonts button in the Business Model Preferences dialog.

The Fonts dialog appears.



3. Select the component you wish to configure from the For drop list.
4. Choose the type of font, System or Portable, from the Kind drop list.
5. Select the typeface, size, and style.
The sample illustrates how your font selection will look.
6. Repeat the process for each business model component you wish to specify.
7. Click the OK button to confirm the new font selections.

Command Summary

File Menu

Command	Description
Properties...	Opens the Business Model Properties dialog.
Supplier Plans...	Opens the Supplier Plans dialog, which shows the supplier plans to the business model, as well as a list of available plans.
Fonts...	Controls the fonts used in the current business model. These settings override the default fonts set by the Workshop Preferences command.
New Application Model...	Creates a new application model. If classes are selected, creates a default application model, including windows based on selected business classes.
Validate Model	Verifies model consistency.
Generate Server Code	Generates the TOOL classes based on the business classes in the business model.
Custom Generation Options...	Opens the Custom Generation Options dialog, which displays the prefix in use for all generated classes. You can use this dialog to change the prefix specification when using custom supplier projects.
Force Complete Generation	Specifies that all classes be regenerated.
Import Database Schema...	Imports database tables into your business model.

Command	Description
Export Database Schema...	Exports the current business model into a file that contains the appropriate SQL create table statements for creating the schema represented by the business model.
Save All	Saves all changes to the workspace.
Workshops Open...	Displays a list of all the workshops that are currently open.
Workshop Preferences...	Allows you to set a variety of characteristics of the Business Model Workshop, such as the size of the drawing area and the components to display.
Print...	Prints the current diagram.
Page Setup...	Invokes a dialog in which you control page and printer settings.
Cancel	Cancels any changes made since the last save.
Close	Saves and closes the workshop.

Edit Menu

Command	Description
Undo	Undoes the last action performed. You can repeat this command as many times as you wish, until you reach the first action you performed in the workshop. This command also undoes any information you added to property dialogs—the entire property dialog is emptied as one action.
Redo	Re-does the last action you reversed using the Undo tool.
Cut	Cuts the selected component to the clipboard.
Copy	Copies the selected component to the clipboard.
Paste	Pastes a cut or copied component from the clipboard.
Clear	Deletes the selected components.
Select All	Selects all the components of the business model.
Find...	Displays a dialog in which you enter the name of a business class. Use this command to locate and display a business class not currently visible in a large business model.

Component Menu

Command	Description
New	Provides a slide-off menu for creating classes, associations, generalizations, and services.
Repeat New	Toggle that allows you to turn Repeat mode for component creation on or off. Alternatively, Ctrl-tool keeps a tool selected until you select another tool
Properties...	Invokes the property dialog of the selected component.
New Attribute...	Creates a new attribute.
Customize...	Invokes the Customization Manager on the currently selected business class.
Class Workshop...	Opens the Class Workshop for the currently selected business class.

View Menu

Command	Description
Toolbar	Toggles on and off the display of the toolbar.
Palette	Toggles on and off the display of the palette.
Service/Class List	Toggles on and off the display of services and classes in the Class List.
Status Line	Toggles on and off the display of the status line.
Attributes	Toggles on and off the display of attributes.
Attribute Keys	Toggles on and off the display of attribute keys.
Attribute Domains	Toggles on and off the display of attribute domains.
Relationships	Toggles on and off the display of associations and generalizations.
Association Labels	Toggles on and off the display of association labels.
Association Arrows	Toggles on and off the display of association arrows.
Grid...	Displays the Grid dialog, where you can choose to display a grid, set grid spacing, and turn on the Snap to Grid property.
Zoom In	Doubles the size of the business model.
Zoom Out	Reduces the size of the business model by 50%.
Zoom...	Displays the Zoom dialog, in which you specify the percentage view to zoom to. Choose a preset value from the radio list, or enter a custom value.

Arrange Menu

Command	Description
Align Classes to	Aligns two or more selected classes to either the top, middle, bottom, left, center, or right class.
Distribute Classes	Evenly spaces the center points of three or more selected classes along a horizontal or vertical axis.
Distribute Connections	Evenly spaces the endpoints of associations and generalizations connected to the selected classes.

Using the Application Model Workshop

Once you have created a business model, you use the Application Model Workshop to define windows based on the business classes in the business model and depict the flow of control between the windows in the application model. This chapter covers:

- using the toolbar and palette
- setting application model properties
- setting default behavior of generated windows
- creating business class windows
- setting window properties
- creating links
- setting link properties
- displaying data in windows
- reusing windows with multiple links
- including existing application models
- calling user-defined methods
- generating client code
- using customized templates

About the Application Model Workshop

You use the Application Model Workshop to develop the windows that form the user interface of your application. You can create one or more windows for each business object in your business model. You can retrieve data from multiple business objects into one window, or display data from each business object in separate windows. The Application Model Workshop provides a set of property dialogs in which you define window types, command interfaces (such as menus, toolbars, and buttons) and links between windows.

The following sections provide an overview of the window characteristics that you can control. More specific information about setting these characteristics is described under the appropriate sections later in this chapter.

Application Model Components

There are three types of components that you can define with the Application Model Workshop:

Business Class Window Always based on a business class, a business class window is analogous to any type of window you create in the Window Workshop.

Included Application Window Allows an application window to call the start window of another application model.

Callout Window Allows the invocation of non-Express windows and methods.

Window Layout

There are three window styles: array field, outline field, or single record form. For information about specifying window layout, see [“Specifying Window Layout” on page 174](#).

Window Behavior

Windows can provide different functionality to the end user, depending primarily on the database command set you specify for the window. A database command is a command that allows a user to manipulate data. For example, a Save command allows a user to update the database. An Insert Record command allows a user to create a new record in the database. The available database command sets are the following:

- All
- Search
- Multiple Record Edit
- Single Record Edit
- Multiple Record View
- Single Record View

See the [“Choosing a Command Set” on page 177](#) for a description of the commands included with each database command set.

For example, if you create a Multiple Record Edit window, then by default, the window will contain appropriate commands for updating multiple records in the database. Similarly, you can create a View window, which will only have commands available for viewing data.

You can also control the type of command interface for each window. An interface is the mechanism for accessing the database commands. The interface options are: Menu and Toolbar, and Buttons.

For more information about controlling window behavior, refer to [“Choosing a Command Set” on page 177](#).

Window Links

Links are the mechanisms through which windows communicate with each other. There are five types of links between business class windows:

- nested
- folder
- drilldown
- command
- lookup

Each of these links defines how a window is invoked. For example, command links always add a button to a window that a user can click on to open another window. Nested links combine two windows to create one window that displays information from two tables—usually in a master-detail relationship.

For more information about using links, see [“Links” on page 185](#) and [“Creating Multiple Links to the Same Window” on page 197](#).

Start Window

The start window for your application determines the control flow of your application. An application begins by invoking the `Display` method on the start window. By default, your start window is the window that has no links into it. If you have more than one window that fits this description, you must explicitly set your start window in the Application Model Properties dialog. See [“Specifying a Start Window” on page 184](#).

Using the Application Model Workshop

There are a number of ways you can access the Application Model Workshop.

- You can select components in your business model and automatically generate a default application model.
- You can create an empty application model from within the Business Model Workshop.
- You can create an empty application model from within the Repository Workshop.

Creating a New Application Model

Whenever you create a new model, the first thing you do is assign it a name.

Application Model Name property The name you give the application model will be the default name for the project generated from the application model. Express appends “Windows” to the application model name to create the project name. For information on the project generated from the application model, see *Customizing Express Applications*. Once you name an application model, you cannot rename it.

► **To create a new application model**

1. From the Repository Workshop, click the New Application Model icon or choose the Plan > New Application Model... command.



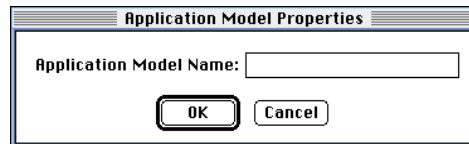
or

1. From the Business Model Workshop, select the classes and associations you wish to include in your application model.

Note that you do not have to select any classes at this point—Express will simply create an empty application model in which you can create your business class windows.

2. Click the New Application Model icon or choose the File > New Application Model... command.

The Application Model Properties dialog appears, as shown below:



3. Enter a name for the application and click OK.

The Application Model Workshop appears, displaying a default application model with default window classes and links for each business class and association that you selected when you gave the New Application Model command. If you have not selected any business classes, you will create windows as described in [“Business Class Windows” on page 172](#).

For a description of the Application Model Workshop elements, see [“The Application Model Workshop Window” on page 162](#). If you are ready to start creating your application model, see [“Business Class Windows” on page 172](#).

Accessing Existing Application Models

To get write access to an application model that has been integrated into the repository, you must either checkout or branch the entire plan. Refer to *A Guide to the iPlanet UDS Workshops* for detailed information about checking out or branching a plan.

- **To modify an existing application model in the Application Model Workshop**
 1. In the Repository Workshop, checkout or branch the application model using the appropriate command on the Plan menu.
 2. In the Repository workshop browser, double-click on the appropriate application model.

The Application Model Workshop appears displaying the application model you selected.

Adding Supplier Plans

Application models allow as supplier plans business models, other application models, and projects.

Every application model must have at least one business model as a supplier plan. If you create an application model by clicking the New Application Model icon from within a business model, Express by default includes that business model as a supplier to the application model.

If you create an application model directly from the Repository Workshop and then drag and drop business classes into the application model, Express automatically includes the business model from which you drag the classes as a supplier plan.

Multiple business models If you use business classes in your application model from multiple business models, then each business model must be a supplier to the application model.

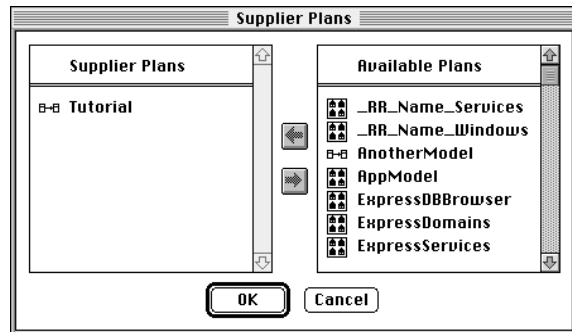
Included application models If your application model includes another application model, then the included application model must be a supplier to your application model. Express automatically includes the included model as a supplier plan.

Callouts If you create a callout by dragging a project that has a start window into your application model, then Express automatically includes the project as a supplier. If you create a callout manually using the New Callout Tool and the callout references a class in another project, then you must explicitly include that project as a supplier to your application model.

In most cases, Express automatically includes the appropriate plans to your application model.

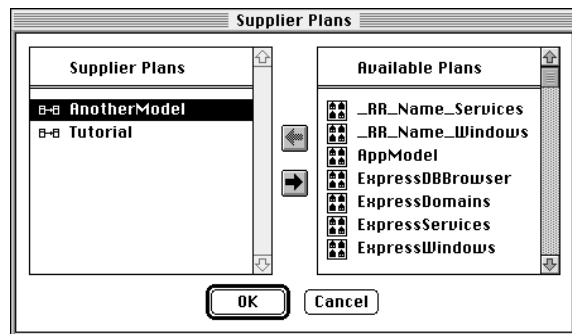
► **To specify a supplier plan for an application model**

1. Choose the File > Supplier Plans... command to display the Supplier Plans dialog.



2. Select the appropriate plan from the Available Plans list and click the left arrow.

The selected plan appears under the Supplier Plans list, as shown below:



3. Repeat until you have transferred all the necessary plans.
4. Click OK.

Supplier Plans and Generated Projects

If you add supplier plans to an application model, Express automatically adds the plan as a supplier to the generated project when it is generated. However, if you remove the supplier plan from an application model, you must manually remove the plan as a supplier to the generated project.

The Application Model Workshop Window

The Application Model Workshop provides the tools you need to create and edit an application model. The Application Model Workshop window, shown in [Figure 6-1](#), consists of the following elements:

Toolbar The toolbar provides a set of tools for editing and aligning graphical elements of the application model.

Palette The palette allows you to create and manipulate an application model. In addition to the Text tool and the Select tool, the palette contains tools for creating windows and links. Types of windows and links are described later in this section.

Floatover Help Floatover help displaying the name of the component appears when you hold the cursor over any application model component.

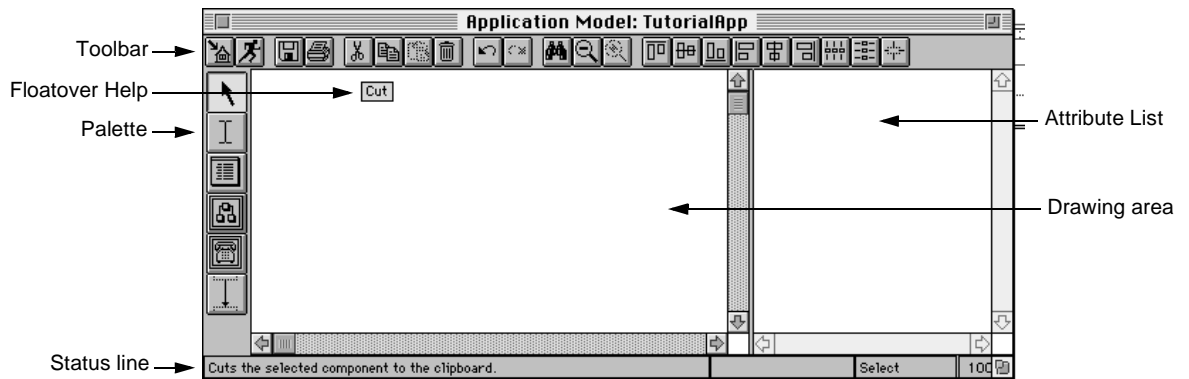
Menus The menus in the Application Model Workshop provide commands that duplicate the functions of the toolbar and palette, as well as provide additional functions, such as printing.

Drawing area The drawing area is the space in which you create an application model.

Status Line The status line is divided into four sections displaying the following information:






- the first section displays the hint text for the icon over which the cursor is positioned
- the second section displays the name of the selected application model component
- the third section displays the tool mode, for example, select or edit
- the fourth section displays the current zoom percentage


Attribute List This Attribute List displays a list of attributes available for display on the selected window.

Figure 6-1 Application Model Workshop Window

Using the Palette

The Application Model Workshop palette provides a set of tools for selecting, editing, creating, and defining windows and links.

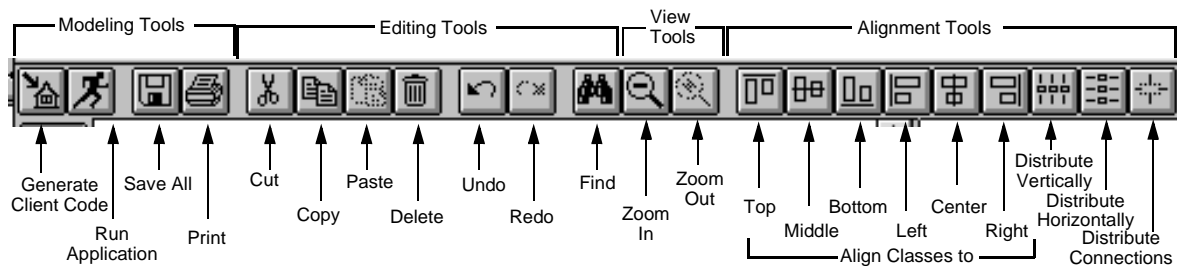
Icon	Tool Name	Description
	Select Tool	Use the Select tool to select and move elements in the application model. Also use the Select tool to open the property dialog of any element of an application model by double-clicking on the element. Select multiple items by drawing a selection box around them or hold down the Shift key while selecting.
	Text Tool	Use the Text tool to enter or edit the name of a window class or link label. When you activate the Text tool, the cursor is an arrow; when you move the cursor over text, it becomes an I-beam and you can enter, delete, or select text as you would in a word processing program.
	New Class Window Tool	Use the New Class Window tool to create a business class window. By default, the window you create is a form window.
	New Included Application Model Tool	Use the New Included Application Model Tool to provide an interface for the start window of another application. You can then call the application from within your application. See “Including an Existing Application Model in the Current Application Model” on page 200.
	New Callout Tool	Use the New Callout Tool to create a mechanism through which an application can invoke external methods or windows directly from generated Express code.

Icon	Tool Name	Description
	New Link Tool	Use the New Link Tool to create a call between windows. Double-click on this link (or select the link and choose the Component > Properties... command) to open the Link Properties dialog, where you define the type of link and its behavior.

Using the Toolbar



The Application Model Workshop toolbar provides a set of tools for editing and formatting an application model. In addition, icons are provided for generating window classes, saving the workspace, and running the application.



Figure 6-2 Application Model Workshop Toolbar Icons



Modeling Tools








Modeling tools are tools you use frequently in creating your model. Most modeling tools are equivalents of File menu commands.

Icon	Tool	Description
	Generate Client Code Tool	Generates TOOL window classes represented by the application model. Equivalent of the File > Generate Client Code command.
	Run Application Tool	Runs the application represented by the application model. If the business classes and/or the window classes are not up to date, Express will generate them before running the application. Equivalent of the File > Run Application command.

Icon	Tool	Description
	Save All Tool	Saves all the changes you have made in your workspace. Equivalent of the File > Save All command.
	Print Tool	Prints an application model, including portions not currently being displayed. If the model is larger than a single page, the Print tool automatically tiles the diagram onto multiple pages, tiling left to right, top to bottom. Equivalent of the File > Print... command.

Editing Tools

Editing tools allow you to manipulate the elements of your application model as you would in any graphical drawing program.

Icon	Tool	Description
	Cut Tool	Removes the selected component and copies it to the clipboard. Equivalent of the Edit > Cut command.
	Copy Tool	Copies the selected component to the clipboard. Equivalent of the Edit > Copy command.
	Paste Tool	Pastes the contents of the clipboard to the location you select. Equivalent of the Edit > Paste command.
	Delete Tool	Deletes the selected component. Equivalent of the Edit > Delete command.
	Undo Tool	Undoes your most recent action. You can undo successive actions as far back as your first action in the workshop since your last save. Actions include entering information in property sheets. Equivalent of the Edit > Undo command.
	Redo Tool	Redoes your most recent action (cancels Undo). Equivalent of the Edit > Redo command.
	Find Tool	Allows you to search for a window class. Equivalent of the Edit > Find... command.

In Select Mode, you can cut or copy the following components:

- windows
- links (along with the windows they connect)

Note that you cannot cut and paste a link independently of the two windows that it connects. Instead, you must select both the windows and the link that connects them before using the Cut tool; the same is true for the Copy tool.

You cannot manipulate attributes with the Cut, Copy, and Paste tools.

Using Cut, Copy, & Paste In Text mode, you can cut or copy text from any part of the diagram onto the clipboard.

➤ **To cut, copy, or delete an element**

1. Select the element in your application model.
2. Select the Cut, Copy, or Delete tool.

➤ **To paste an element**

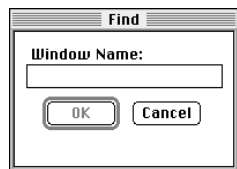
1. Cut or copy an element using the method described above.
2. Select the Paste tool.
3. Click in the window where you want to paste the element.

Using the Find tool The Find tool is especially useful if your model is large and you cannot see all the classes in the drawing area at one time.

➤ **To find a window class**

1. Click the Find tool.

The Find dialog appears.





2. Enter the name of the class you wish to find and click OK.

Express highlights the window class (and brings it into view if it is not currently in the drawing area), and highlights the class name in the Attribute List.

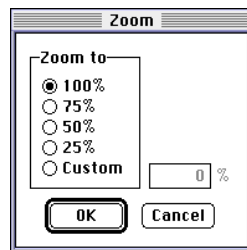
Viewing Tools

The View tools include the toolbar Zoom tools, the View > Zoom... command, and the View > Grid... command. The toolbar tools are shown in the following table; the command descriptions follow the table.

Icon	Tool	Description
	Zoom In Tool	Doubles the size of the current application model. This tool only affects a model that is being displayed at less than 100%. Equivalent of the View > Zoom In command.
	Zoom Out Tool	Reduces the size of the current application model. By default, each time you select this tool, the view reduces by 50%. Equivalent of the View > Zoom Out command.

Using the Zoom... command The Zoom... command does not have a toolbar equivalent. To zoom to a specific percentage view of the application model, choose the View > Zoom... command. The Zoom dialog appears, as shown in [Figure 6-3](#).

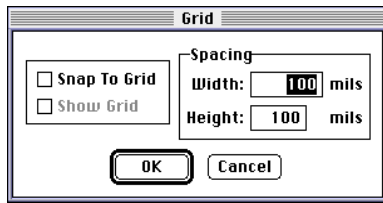
Figure 6-3 Zoom Dialog



Click one of the radio buttons to zoom to a preset percentage (100%, 75%, 50%, 25%) and then click OK. You can also click the Custom radio button and then enter a specific percentage in the field to the right and then click OK.

Using the Grid... command The Grid... command does not have a toolbar equivalent. You can modify alignment grid properties using the View > Grid... command. Choose the View > Grid... command and the Grid dialog appears, as shown in [Figure 6-4](#).

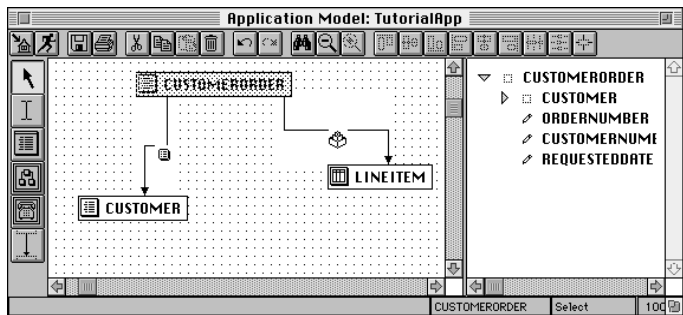
Figure 6-4 Grid Dialog



Snap To Grid property Turn on the Snap To Grid toggle to subsequently align application model components with grid lines. This property will affect only those components you create after you turn on the toggle.

Show Grid property The Show Grid property becomes active if you turn on the Snap to Grid property. Turn on the Show Grid property to display the grid lines in the drawing area, as shown [Figure 6-5](#).

Figure 6-5 Grid Lines




Spacing To specify the size of the grid, enter the Width and Height in mils.

Alignment Tools

The alignment tools help format application models.

Icon	Command	Description
	Align Windows to Top Tool	Aligns the tops of two or more selected windows to the top-most window. Equivalent to the Arrange > Align Windows to Top command
	Align Windows to Middle Tool	Aligns two or more windows through the vertical middles of the windows. Equivalent to the Arrange > Align Windows to Middle command
	Align Windows to Bottom Tool	Aligns the bottoms of two or more windows to the bottom-most window. Equivalent to the Arrange > Align Windows to Bottom command
	Align Windows to Left Tool	Aligns the left edges of two or more windows to the left-most window. Equivalent to the Arrange > Align Windows to Left command
	Align Windows to Center Tool	Aligns two or more windows through the horizontal center of each window. Equivalent to the Arrange > Align Windows to Center command
	Align Windows to Right Tool	Aligns the right edges of two or more windows to the right-most window. Equivalent to the Arrange > Align Windows to Right command
	Distribute Windows Horizontally Tool	Evenly spaces three or more selected windows along a horizontal axis. Equivalent to the Arrange > Distribute Windows Horizontally command
	Distribute Windows Vertically Tool	Evenly spaces three or more selected windows along a vertical axis. Equivalent to the Arrange > Distribute Windows Vertically command

Icon	Command	Description
	Distribute Connections of Windows Tool	Evenly spaces the endpoints of links connected to selected windows. Equivalent to the Arrange > Distribute Connections command

► **To use an alignment tool**

1. Select the appropriate number of windows (see table above).
2. Select the alignment tool.

Viewing Workshop Components

You can select the components of the application model you wish to view by using the commands on the View menu. By default, your workshop preferences specify which components are visible. You can either change the default settings (see [“Setting Workshop Preferences” on page 215](#)) or select the components you wish to view or hide from the View menu. A check mark next to the component indicates it will be displayed. To hide a component, select its menu command again.

Controlling Fonts

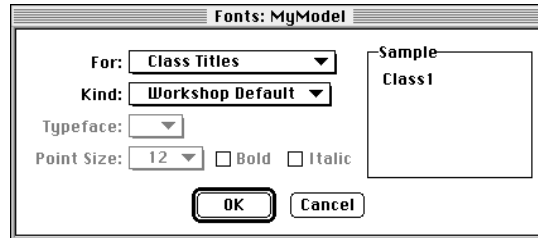
The File > Fonts... command lets you control the fonts used in the current application model. The fonts settings will override the default fonts set in the Workshop Preferences dialog. The File > Fonts... command displays a dialog with the following properties:

Property	Description
For	The For property specifies the area for which you are specifying the font. You can set the font for application class window titles, links, Callout titles, and Included Application titles.
Kind	The Kind property specifies the type of font: Workshop Default (specified in the Workshop Preferences dialog), portable, or system.
Typeface	The Type property specifies the typeface.
Point Size	The Size property specifies the size of the font.
Bold/Italic	The Bold and Italic toggles specify whether the font is bold or italic.

► **To set fonts for the current application model**

1. Choose the File > Fonts... command.

The Fonts dialog appears.



2. From the For droplist, choose the area for which to set the font.
3. From the Kind droplist, choose the type of font to set.
4. Turn on the Bold and/or Italic toggles, if necessary.
5. Click OK.

Access to Other Workshops

By default, the TOOL classes that Express generates from your application model are read only. You can create a customizable class for an individual class or you can specify that Express automatically create the full hierarchy for every class. Once customizable business classes exist, you can enter the Class Workshop for a selected class with the Component > Class Workshop command.

For complete information about creating customizable Express classes, see *Customizing Express Applications*.

Leaving the Application Model Workshop

To exit the Application Model Workshop, use the File > Close command. You can also use the File > Cancel command, which closes the workshop and discards any changes you have made since the last save.

Business Class Windows

You build an application model by defining windows that provide the interface to your application.

A business class window in the Application Model represents a window in an application; each business class window is based on a business class in a business model. You can base multiple windows on the same business class. You can also use several business models to create a single application.

Express gives you complete control over the appearance and behavior of application windows. The Business Class Window Properties dialog allows you to set a variety of properties that control the appearance and behavior of a window. These properties are described in the following sections.

In addition to settings for window layout, the Business Class Window Properties dialog also provides settings to control the types of commands that are available on the window, as well as the type of interface to the commands. For example, you can specify that the window is for viewing a single record—no edit or update commands will appear on the window. If you specify that the window is for editing multiple records, the appropriate commands, such as insert, delete, and scrolling, will appear on the window. You also have the option of specifying the command interface, such as menu and toolbar or buttons.

Creating a Business Class Window

There are several ways to create a business class window.

► **To create a window class by dragging a Business Model class into the Application Model Workshop**

1. Open an existing business model and open the Application Model Workshop.

Remember, you can open an empty application model by clicking the New Application Model icon from the Business Model Workshop or the Repository Workshop.



2. Select a business class name from the Business Class/Service List in the Business Model workshop.

3. Drag the business class name into the Application Model Workshop.

Note that you cannot drag a service into the Application Model Workshop.

Using multiple business models You can create application models using business classes from multiple business models by opening each business model and dragging the appropriate class or classes into the application model. Refer to [“Associations and Links” on page 189](#) for information about drawing links between classes from different business models.

➤ **To create a window class using the Application Model Workshop palette**

1. Open the Application Model Workshop.
2. Select the New Class Window tool.



3. Click in the window area.

Alternatively, you can create a window using the New Class Window... command under the New submenu on the Component menu.

The Window Class Properties dialog appears.

4. Enter the name of a business class from the supplier business model on which to base the business class window.

➤ **To create multiple window classes without reselecting the New Class Window tool**

1. Choose the Component > Repeat New command.

The workshop remains in Repeat New mode until you toggle the command off. This means that you can create multiple versions of any component while in Repeat New mode.




2. Select the New Class Window tool.



3. Click once in the window area for each class you need.

Specifying Window Layout

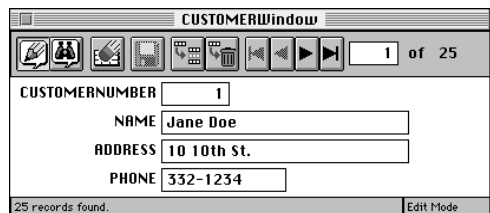
There are three window layout styles: form, outline, and array. You specify the window layout using the Layout of Fields property in the Business Class Window Properties dialog (see [“Setting Window Properties” on page 180](#)). In most cases, the default window layout is Form. Each window layout type has an associated icon that appears on the window class in the application model. The icons are:

Icon	Command	Description
	Form	Displays one record at a time. For the layout, you can specify the direction of the fields (horizontal or vertical) and the number of fields to wrap, that is, the number of fields that fit in the window before wrapping to the next column or row. The default direction of the fields is Down.
	Array	Displays all retrieved records in an array field. Displays records in an array field. By default, the array field has scroll bars and its own toolbar commands for manipulating the array.
	Outline	Displays records in an outline field. On Windows95/NT machines, outline fields look like list views and support runtime column resizing and sorting.

Window Style

Form window A form window displays one record at a time, as shown in [Figure 6-6](#) below. For form style windows, you can control characteristics such as the number of fields to wrap and the direction to place the fields.

Figure 6-6 Form Window



Array window An array window displays records in an array field. [Figure 6-7](#) below displays an array window:

Figure 6-7 Array Window

CUSTOMERNUMBER	NAME	ADDRESS	PHONE
1	Jane Doe	10 10th St.	332-1234
2	J. Smith	11 11th Ave.	332-4321
3	R. Simmons	111 A St.	443-4321
4	CUSTOMER 4	ADDRESS FOR CUSTOMER 4	510-777-2222
40	Roebuck Staples	River Road	888

Outline window An outline window displays all retrieved records, one record per line in an outline field. An example of an outline window is shown in [Figure 6-8](#).

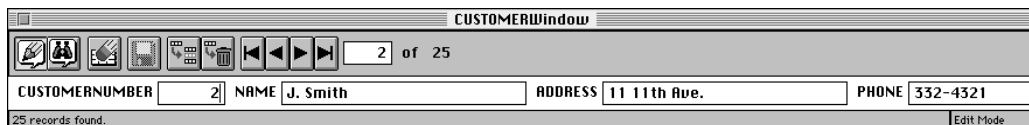
Figure 6-8 Outline Window

CUSTOMERNUMBER	NAME	ADDRESS
1	Jane Doe	10 10th St.
2	J. Smith	11 11th Ave
3	R. Simmons	111 A St.
4	CUSTOMER 4	ADDRESS FOR
40	Roebuck Staples	River Road
56	Walter Neth	Burbank
144	gfdgd	ffds
555	Monsieur Baggins	River Road
708	Gram Parsons	River Road

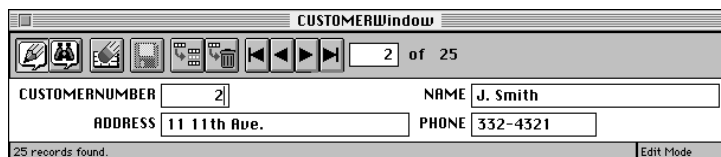
Field Placement

If the Layout of Fields property is set to Form, there are two other properties that affect window display: Direction and Wrap.

Direction property The Direction property allows you to place fields on a window in a horizontal or vertical direction. By default, the Direction property is Down, as is shown above in [Figure 6-7](#). If you set the Direction property to Across, then you will get a window that looks like the one in [Figure 6-9](#):

Figure 6-9 Direction Property Set to Across

Wrap property You can specify how many fields to place in a row/column before wrapping to the next row/column by setting the Wrap property. For example, in [Figure 6-9](#), the fields are placed across the window (in a row); if you set the Wrap property to 2, you'll get a window that looks like [Figure 6-10](#):

Figure 6-10 Wrap Property Set to 2

Choosing a Command Interface

A window's interface is what gives a user access to an application's commands.

Default Interface property The Default Interface property lets you specify one of the following command interfaces:

Buttons The Buttons command interface provides all the commands available to the window as buttons.

Menu and Toolbar The Menu and Toolbar command interface provides a menu bar and a toolbar to access the commands available to the window.

Every command is available on both interfaces.

You need to think about the type of window you are building as you decide what kind of command interface to use. For example, if you are creating a window in which to browse customer records, you probably want to use a menu and toolbar interface so the user has easy access to the scrolling commands. If you are creating an informational dialog, you probably will want to use a button interface, which supplies an OK button to dismiss the window. Remember, though, that you can always customize a window to add, for example, specific buttons to a window using a menu command interface.

Note that if you use the Adaptive Command Set property (see [“Creating Flexible Windows” on page 179](#)), Express disregards the Default Interface setting for all windows except the application’s start window.

Choosing a Command Set

In addition to determining the type of interface for each window, you can also specify the set of commands that are available on a window. For example, if the window is to be used for data entry, then choose the Multiple Record Edit option to make available the appropriate Insert, Update, Save, and Delete commands. If you are building an informational dialog, then the Single Record View option is appropriate, because that kind of window does not need any type of data manipulation commands.

Note that Express disregards this property if you use the Adaptive Command Set property. See [“Creating Flexible Windows” on page 179](#) for more information.

Command Set property The following tables list the commands available for each command set, for each command interface.

Menu Commands	All	Multiple Edit	Single Edit	Search	Multiple View	Single View
File Menu	●	●	●	●	●	●
Preferences	●	●	●	●	●	●
Revert	●	●	●			
Print	●	●	●	●	●	●
Print Setup	●	●	●	●	●	●
Save	●	●	●			
Cancel	●	●	●	●	●	●
Close	●	●	●	●	●	●
Exit All	●	●	●	●	●	●
Edit Menu	●	●	●	●	●	●
Result Set Menu	●	●		●	●	
Mode	●					
Clear	●			●		
Search	●			●		

Menu Commands	All	Multiple Edit	Single Edit	Search	Multiple View	Single View
Insert	●	●				
Delete	●	●				
Scroll	●	●		●	●	

Toolbar Commands	All	Multiple Edit	Single Edit	Search	Multiple View	Single View
Mode	●					
Clear	●			●		
Save	●	●	●			
Search	●			●		
Insert	●	●				
Delete	●	●				
Scroll	●	●		●	●	

Button Commands	All	Multiple Edit	Single Edit	Search	Multiple View	Single View
Insert		●				
Delete		●				
OK		●	●	●	●	●
Clear				●		
Search				●		
Cancel		●	●	●	●	●
Scroll		●		●	●	

Command Sets and Outline Windows

Although the Single Record Edit and Multiple Record Edit command sets are available for outline windows, users cannot directly edit data displayed in a window that displays an outline field. The edit command sets are useful when an outline window is linked to other windows.

Editing records For example, to allow the end user to edit data in an outline field, you must create a second window based on the same table as the outline window and set its Layout of Fields property to Form. Create a drilldown link from the outline window to the form window. Now, when the user wishes to edit a record, he or she must double-click on the record, bringing up a form window displaying the same data. Edits can be made, and Express will pass the changes back to the outline window when the user closes the called window.

Inserting records In addition, to allow the end user to insert or delete records in an outline field, you must specify the Buttons default interface. The Buttons default interface will provide insert and delete push-button. To insert a new record, the user clicks the Insert button, and Express adds a blank line at the bottom of the outline field. The user double-clicks on the blank line, invoking a form window where the user can enter the new record. The data is passed back to the outline window when the user closes the called window.

Creating Flexible Windows

As described above, you can explicitly tell Express which commands and what type of interface to use on any particular window. However, there may be times when you will want to use the same window in a variety of situations. For example, you might want to give some users of your system the ability to edit data, and others the ability to only view data. To do this, you might want the same window to appear with different command sets. To reuse a window, you simply draw another link to it (see [“Links” on page 185](#) for more information about creating links). However, you can specify only one type of command set unless you use the Adaptive Command Interface property.

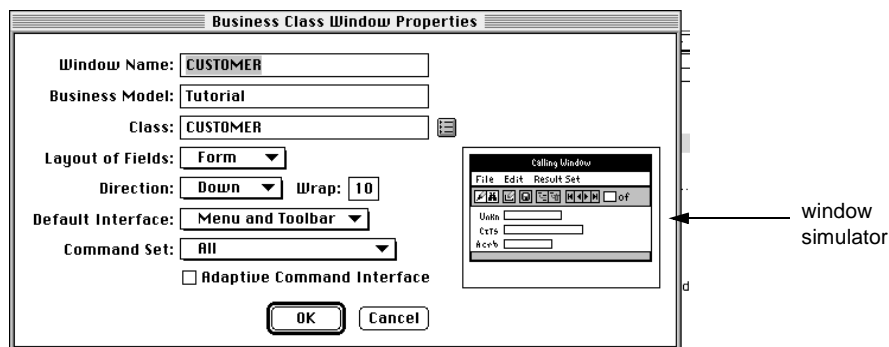
Adaptive Command Interface property Express is designed to provide the most appropriate and logical command set and interface based on the type of windows you are creating and the association between the underlying business classes. The Adaptive Command Interface property tells Express to determine the appropriate command set and interface—when you use this property, the resulting command set and interface will override any explicit settings you made using the Default Interface and Command Set properties. You should set the Adaptive Command Interface property only if you plan to draw multiple links to the same window. For more information, see [“Creating Multiple Links to the Same Window” on page 197](#).

Setting Window Properties

► To set window properties

1. Double-click on the window whose properties you wish to set, or select the window and choose the Component > Properties... command.

The Business Class Window Properties dialog appears, as shown in the figure below.



Note the window simulator. This image displays an approximation of the generated window, including field layout, command interface, and command set. The window simulator will change to reflect changes you make to window properties.

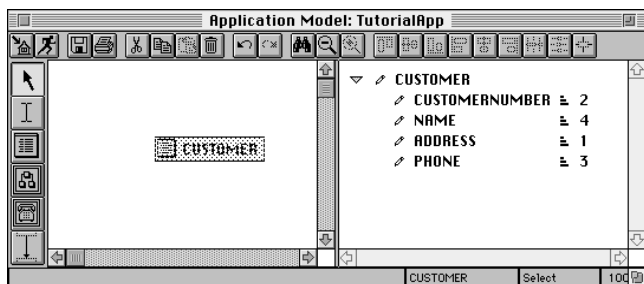
2. Set the appropriate properties and click OK.

Editing Window Properties

At any time, you can modify the settings for an application window by double-clicking on the window, or selecting the window and choosing the Component > Properties... command. In the subsequent Business Class Window Properties dialog, make the necessary changes and click the OK button.

Specifying Window Fields in the Attribute List

Every business class window is associated with a database table. You can determine which business class attributes to display as fields on generated windows. When you select a window class with the Select tool, the attributes that belong to the business class on which the window is based are shown in the Attribute List area, as shown in [Figure 6-11](#).

Figure 6-11 Attribute List

Displaying fields By default, each attribute in the list will appear as a field on the generated window. Note, however, that if you add an attribute to a business class, the corresponding window class attribute will not be selected to appear by default—you will have to select it in the attribute list.

Attributes that will appear on a generated window will have a pencil, or a pencil with a slash next to them.

If you do not want an attribute to appear as a field on the generated window, click the pencil until an empty box appears. The empty box means that the attribute will not be displayed on the generated window.

Writable and read-only fields A pencil next to an attribute indicates the field on the generated window will be writable. If you want to display a field in read-only mode, click the pencil and a slash appears through it, indicating that the field will be read only.

Sort direction The icon to the right of the attribute name indicates the sort direction of the data displayed in the field. If the icon is dimmed, then no sorting is specified. To sort the data in ascending order, click the icon until the bars appear bold and smallest to largest. To sort the data in descending order, click the icon until the bars appear bold and largest to smallest.

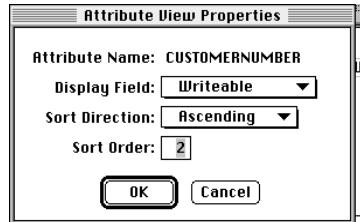
Sort order If you select a sort direction, a number appears to the right of the attribute. The number indicates the priority in which the field will be sorted. For example, if you have FirstName and LastName attributes, it would make sense to sort the LastName field first, and then the FirstName field. The first attribute you give a sort direction to will be numbered 1, the second 2, and so on.

An alternative to setting the properties using the graphical icons is to use the properties dialog.

➤ **To set attribute view properties**

1. Double-click on an attribute.

The Attribute View Properties dialog appears, as shown below:



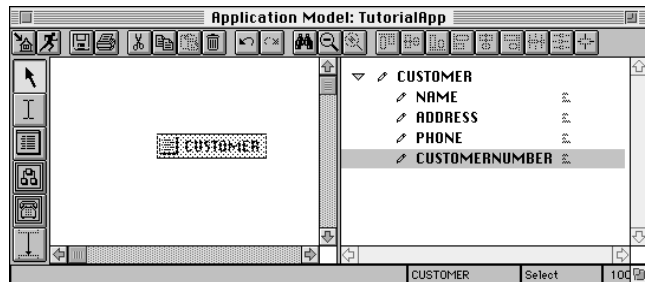
2. Set the appropriate properties.
3. Click OK.

Changing Field Order

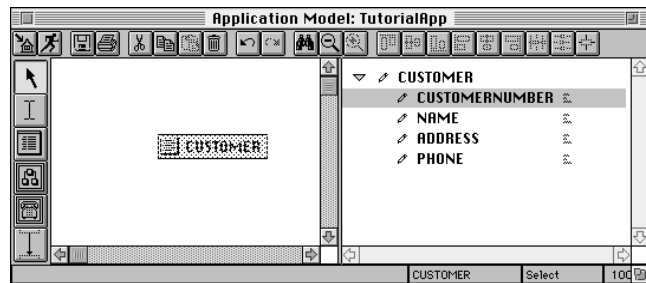
You can change the order fields are displayed on a generated window by changing their order in the Attribute View List.

➤ **To change field order**

1. Select the field you wish to move, as shown below.



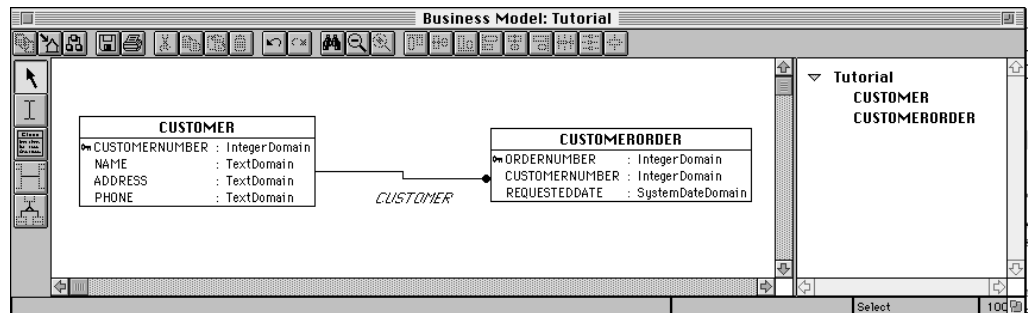
2. Drag the field to the new position, as shown below.



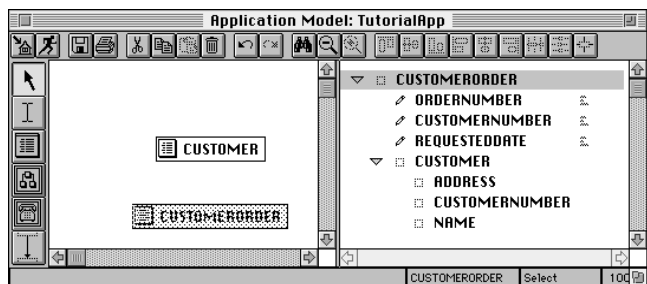
Specifying Window Fields from Two Business Classes

If you have a to-one association in a business model and create a business class window based on the from-class, then the attributes on the to-class will be available for display on the generated window. For example, [Figure 6-12](#) displays a many to one association between CustomerOrder and Customer.

Figure 6-12 Many to One Association Between CustomerOrder and Customer



In the Attribute View List in the Application Model Workshop, the Customer attributes appear in the CustomerOrder outline field, as shown in [Figure 6-13](#).

Figure 6-13 CustomerOrder Attribute View List Contains Customer Attributes

You can display the Customer attributes just as you would the CustomerOrder attributes, except the Customer attributes can be displayed only in read-only mode.

Re-ordering To-Class Attributes To place to-class attributes before the from-class attributes on a generated window, you must move the entire to-class outline field. See [“Modifying Field Order” on page 64](#).

Changing Field Width

You specify the width of a field in the Business Model Workshop. See [“Field Label Text and Width” on page 113](#).

Specifying a Start Window

To start an application, you need to provide an entry point. To do this, you specify a start class. Because an application model consists of windows, the start class will always be a window class. Express begins execution of the application by creating an object of the start class and then invoking the start method on it. Since the start class is a window class, the start method is the Display method, which initializes the data for the window, opens the window, and handles the window’s events.

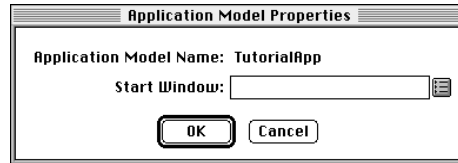
iPlanet UDS assigns as the default start window the window class in your application model that is not called by another window class. See [“Links” on page 185](#) for information about called and calling window. Note that as you add windows to your application, Express will try to determine which window is the start window, but you should double-check before you run the application.

Start Window property Application models have a property called Start Window in which you specify the start window for an application.

► **To specify a start window**

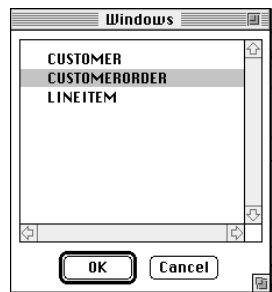
1. Choose the File > Properties... command.

The Application Model Properties dialog appears, as shown below.



2. Enter the name of the start window.

You can also select a window class by clicking the browser button, which displays the dialog shown below. Choose a class and click OK, or double-click on a class.



3. Click OK.

Links

Links are the mechanisms through which one window invokes another window and, along with windows, are the building blocks of an application. The end user typically activates a link by clicking a button (command link, lookup link), double-clicking in an outline field (drilldown link), or selecting a folder tab (folder link). Some links display both windows automatically, without any explicit user action (nested link).

Calling window, called window The window on which you perform one of these actions is referred to as the “calling window.” The resulting window is referred to as the “called window.” Windows connected by nested and folder links appear in one window, and therefore there is no mechanism with which to invoke the called window—it appears automatically.

Creating a Link

When you create a link, it is by default a Command link. Once you create a link, you can change its type as well as set a variety of properties that affect the behavior of your application. Properties of links are discussed in the following sections. You set link properties in the Link Properties dialog, as described in [“Setting Link Properties” on page 194](#).

► To create a link




1. Select the New Link tool.





2. Place the cursor in the “calling” window, hold down the mouse button, drag the cursor into the called window, and release the mouse button.

Link Types

The five types of links are described in the table below:

Link Icon	Link Type	Description
	Nested	Always creates a relationship where the called window is displayed nested within the calling window. Use to view multiple tables in one window. You can create nested links between all types of windows.
	Command	Provides a button on the calling window that, when clicked, invokes the called window as a separate window. Can also be used as a link to a callout window.
	Drilldown	Allows a user to double-click in a field in the calling window to invoke a called window as a separate window. Only used for outline fields.

Link Icon	Link Type	Description
	Folder	Displays related data in a series of tabbed folders in the same window. You can nest multiple windows within one calling window; each called window is contained in its own folder.
	Lookup	Creates a button on the calling window that, when clicked, brings up the called window in a separate window. You can use the called window to retrieve information, which is then passed back to the calling window.

The type of link you can use depends on the window type of the calling window. The following table describes the links available for each window type:

Window Type	Available Links
Form	Command, Lookup, Nested, Folder
Outline	Command, Drilldown, Nested, Folder
Array	Command, Lookup, Nested, Folder

Type property You specify a link type by selecting an option from the Type property drop list on the Link Properties dialog.

Label property Command links use buttons to invoke the called window. You specify the text for the button labels in the Label property field on the Link Properties dialog. Also, use this property to specify the tab text for folder links.

Determining Data to Display in Called Window

When you link two windows, Express gives you options for specifying what data, if any, the called window should display when it is initially invoked. For example, if you create a lookup link, you can specify that the called window display all the records from the table on which it is based, or display no records. If you specify that the lookup window display all records, the user can scroll through the records until the appropriate record is selected, and then close the window, which causes the selected record to be passed back to the calling window. If you specify that the lookup window display no data, then the user can perform a query to retrieve the appropriate record, and pass it back to the calling window.

Display property The Display property on the Link Properties dialog is a drop list containing some or all of the data display options. The drop list is dynamic—that is, different options are available depending on the types of links you are using and the relationship between the business classes on which the windows are based. The Display options, their descriptions, and when they are available are listed in the following table:

Link Type	Display Option	When Available	Called Window Display
Command	No records	Linking any type of window.	No records are displayed in the called window.
	All records in table	Linking any type of window.	Displays all records in the table on which the called window is based.
	Associated	Linking windows based on tables that have an association.	Displays the records associated with the current record in the calling window.
	New record (associated)	Linking windows based on tables that have a “to-one” association.	Displays a new (empty) record, which is inserted into the table on which the called window is based, and associated with the table on which the calling window is based.
	Selected	Linking windows based on same table.	Displays the selected record in the calling window.
	All records in window	Linking windows based on same table.	Displays all the records in the calling window.
	New record (insert)	Linking windows based on same table.	Displays a new (empty) record in the called window, which is inserted into the table on which the window is based when the user closes the window.
Drilldown	Associated	Linking windows based on tables that have an association.	Displays the records associated with the current record in the calling window.
	Selected	Linking windows based on same table.	Displays the current record in the calling window.

Link Type	Display Option	When Available	Called Window Display
Folder	All records in table	Linking any type of window.	Displays all records from the table on which the called window is based.
	Associated	Linking windows based on tables that have an association.	Displays the records associated with the current record in the calling window.
	Selected	Linking windows based on same table.	Displays the selected record in the calling window.
Lookup	All records in window	Linking windows based on same table.	Displays all the records in the calling window.
	No records	Linking windows based on tables that have a to-one association.	No records are displayed
Nested	All records in table	Linking windows based on tables that have a to-one association.	Displays all records in the table on which the window is based
	All records in table	Linking any type of window.	Displays all the records from the table on which the called window is based.
	Associated	Linking windows based on tables that have an association.	Displays the records associated with the current record in the calling window
	Selected	Linking windows based on same table.	Displays the selected record in the calling window.
	All records in window	Linking windows based on same table.	Displays all the records in the calling window.

Associations and Links

The types of relationships you can build between windows in the Application Model Workshop are dependent on the association between the business classes in the business model on which the windows are based:

- You can build a master-detail relationship between two windows if the business classes on which the windows are based have a one-to-many or many-to-one association. The records in the detail window will not be editable.

- You can build a master-detail relationship in which the detail records are editable if the business classes on which the windows are based have a one-to-many aggregate association.
- You can build nested windows that are not in a master-detail relationship if the business classes on which the windows are based have a many-to-one or one-to-one association.
- You can build a lookup link between two windows if the business classes on which the windows are based have a many-to-one or one-to-one association.

Association property The Association property specifies the association in the supplier business model on which the link is based. Typically, a link connects two windows that have a relationship in the business model. In such a case, the drop list will display that association. Some classes may have multiple associations between them. In these cases, all associations will appear in the Association drop list. Choose the association on which you wish to base your link—make your decision based on the scenarios described above.

Links between classes from different models An application model that uses windows from different business models will treat the windows as if they were created from unassociated business classes from the same business model. In other words, when you link the windows, the Association property is always empty.

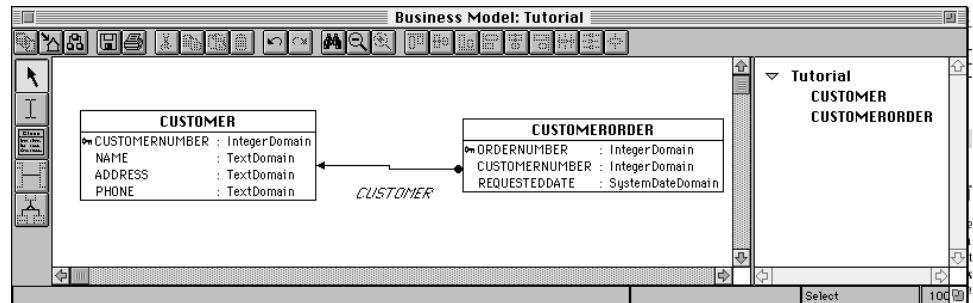
Fetching Detail Data

Often, the data you are displaying in windows you create with Express will reflect a master-detail relationship. That is, the data retrieved from one table is based on unique criteria from another table. For example, retrieving all the orders placed by a particular customer requires a master-detail relationship. Similarly, retrieving all the line items for a single order is another example.

Pre-Fetch Dependent Records property If you are creating a link that is part of a master-detail relationship, Express provides an option that can alter the performance of the query. The Pre-Fetch Dependent Records property tells Express to retrieve all detail records along with the master records in the initial query. As the user scrolls through the master records, the corresponding detail records are displayed from this result set. If this property is not selected, then Express retrieves only the first detail record. Each time the user scrolls to the next master record, Express issues a query to the database to retrieve the appropriate detail records. In general if you use this property, Express may take longer to bring up the initial window, but scrolling through the records will be faster. Conversely, if you do not use this property, the initial window may appear faster, but each scroll command will execute a fetch to the database, and may perform less quickly.

Association direction Note, however, the direction of the association on which the link is based affects whether or not the Pre-Fetch Dependent Records property is available. (See “[Direction](#)” on page 118 for information about associations.) For example, in the master-detail relationship between Customer and CustomerOrder, the underlying association looks like the figure below:

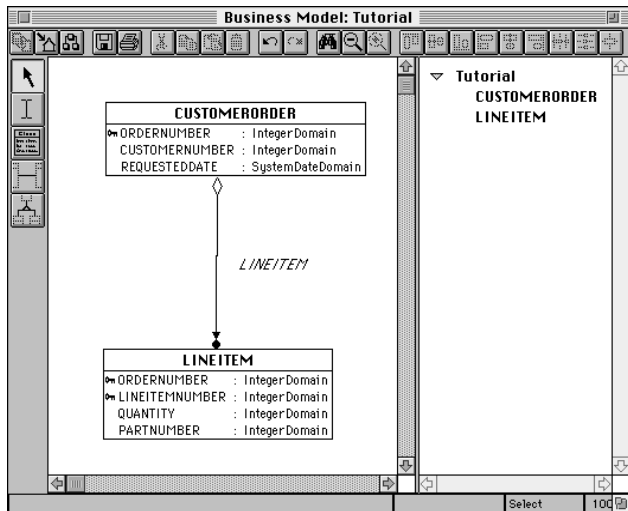
Figure 6-14 CustomerOrder to Customer—Many-to-One Association



There is a many-to-one relationship from CustomerOrder (from-class) to Customer (to-class). In the application model, the master record is based on a Customer object and the detail records are CustomerOrder objects. Because Customer objects do not have pointers to CustomerOrder objects (Customer is the to-class), Express cannot pre-fetch detail data. The fetch occurs each time a new master record is selected.

Conversely, in the CustomerOrder to LineItem example, the underlying association looks like the following:

Figure 6-15 CustomerOrder to LineItem—One-to-Many Association



There is a one-to-many association from CustomerOrder to LineItem. In the application model, the master record is based on a CustomerOrder object and the detail records are LineItem objects. Because CustomerOrder objects have points to LineItem objects (LineItem is the to-class), Express can pre-fetch all dependent data for each master record.

Fetching columns The Pre-Fetch Dependent Records property also applies to columns in a single record. For example, if your application passes data between windows based on the same class, and different fields appear on each window, the Pre-Fetch Dependent Records property tells Express to retrieve data for fields displayed on either window.

Returning Data From Lookup Links

If you wish to display associated information on return from a lookup link, then the window called by the lookup link must display the fields you want to display on the calling window. Otherwise, these fields will be blank until the parent record is re-fetched from the database.

Called Window Properties

There are two characteristics of windows that are controlled by the Link Properties dialog: read status and mode status.

Read/Write window The read status drop list determines if the window is read-only or read/write.

Modal/Modeless window The mode status drop list determines if the window is modal or modeless. Modal windows are synchronous—in other words, the user must close the called window before another window can be active. For example, lookup windows are always modal.

A modeless window is asynchronous—other windows can be active at the same time. Modeless windows do not return data. If you are creating a folder or nested link, the mode status drop list will be empty.

Note that if you selected the Adaptive Command Interface property for the window on the Window Class Properties dialog and you select Modal in the Link Properties dialog, the called window will use a button interface. Express uses the Buttons interface because a modal window is often a dialog. If you selected the Adaptive Command Interface property and you select modeless, the called window will use a menu and toolbar interface. Express uses the Menu and Toolbar interface because a modeless window is often a main window.

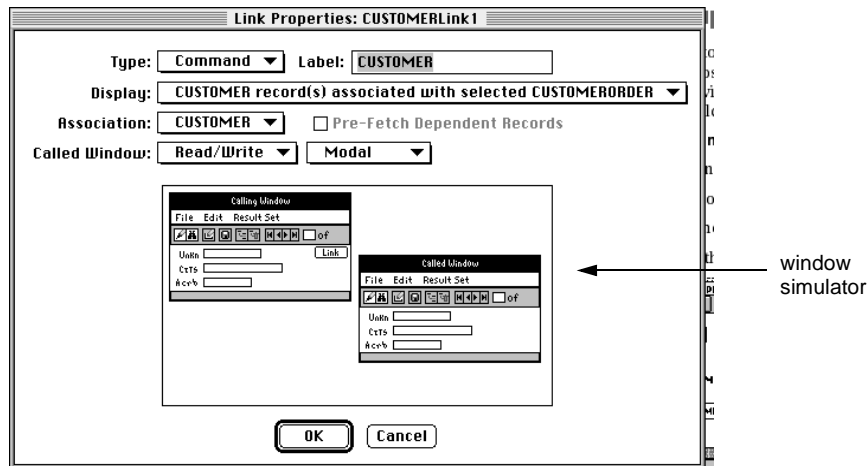
Setting Link Properties

You set link properties in the Link Properties dialog.

► To set link properties

1. Double-click on the link.

The Link Properties dialog appears, as shown below.



The window simulator in the Link Properties dialog displays a general representation of the calling and called windows. Each window is shown with the commands that will be generated based on the current settings in the dialog. As you change the settings, the simulator will adjust to show the affect of the change on the generated windows.

2. Set the appropriate properties and click OK.

About Nested Links

You can create nested links between any types of windows; each window can be a different type, for example [Figure 6-16](#) shows a form nesting an outline, or the same types—[Figure 6-17](#) shows an array nesting an array.

Figure 6-16 Form Window Nesting an Outline Window

The screenshot shows a window titled "CUSTOMERWindow" with a toolbar and a status bar. The main area contains a form with the following fields:

- CUSTOMERNUMBER:
- NAME:
- ADDRESS:
- PHONE:

Below the form is a table with the following data:

ORDERNUMBER	CUSTOMERNUMBER	REQUESTEDDATE
0	1	25-Apr-1997 00:00:00
2	1	25-Apr-1997 00:00:00
26	1	19-Nov-1996 01:00:00
150	1	28-Mar-1997 16:08:21
152	1	21-Mar-1997 16:31:39
232	1	25-Nov-1996 01:00:00
1001	1	01-Jan-1995 01:00:00
1002	1	02-Jan-1995 01:00:00
123453	1	14-Mar-1997 00:00:00

The status bar at the bottom indicates "25 records found." and "Edit Mode".

Figure 6-17 Array Window Nesting an Array Window

The screenshot shows a window titled "CUSTOMERWindow" with a toolbar and a status bar. The main area contains a table with the following data:

CUSTOMERNUMBER	NAME	ADDRESS	PHONE
1	Jane Doe	10 10th St.	332-1234
2	J. Smith	11 11th Ave.	332-4321
3	R. Simmons	111 R St.	443-4321
4	CUSTOMER 4	ADDRESS FOR CUSTOMER 4	510-777-2222
40	Roebuck Staples	River Road	888

Below this table is another table with the following data:

ORDERNUMBER	CUSTOMERNUMBER	REQUESTEDDATE
543	2	22-Nov-1996 01:00:00
1003	2	03-Jan-1995 01:00:00
1004	2	18-Nov-1996 01:00:00

The status bar at the bottom indicates "25 records found." and "Edit Mode".

Making Nested Windows Updateable

When you build the business model that contains the classes for the nested relationship, you must include an association from the parent class to the nested class. The association can be either one-to-many or one-to-one. (Remember, you can have multiple associations between the same classes without conflict.)

In the application model that defines the windows used in the nested relationship, be sure to set the called window to Read-Write in its Link Properties dialog, and set the Association property to use the association that points to the nested class.

Button interface and Multiple Record Edit command set To insert data into nested array or outline windows, you must select the Buttons Default Interface property (see “Choosing a Command Interface” on page 176) and the Multiple Record Edit command set (see “Choosing a Command Set” on page 177) for the nested window. Express will generate Insert and Delete buttons, as shown in Figure 6-18.

Figure 6-18 Form Window Nesting an Outline Window With a Button Interface

The screenshot shows a form window titled "CUSTOMERORDERWindow". At the top is a toolbar with various icons and a "4 of 10" indicator. Below the toolbar are three input fields: "ORDERNUMBER" with the value "1001", "CUSTOMERNUMBER" with the value "1", and "REQUESTEDDATE" with the value "1/1/95 12:00:00 am". Below these fields is a table with three columns: "LINEITEMNUMBER", "QUANTITY", and "PARTNUMBER". The table contains two rows of data: the first row has "1", "3", and "90001"; the second row has "2", "1", and "90002". Below the table are three buttons: "Insert", "Delete", and "1 of 2". At the bottom of the window, it says "10 records found." and "Edit Mode".

LINEITEMNUMBER	QUANTITY	PARTNUMBER
1	3	90001
2	1	90002

Nested array When the user clicks the Insert button, Express inserts a blank line into array field. Enter the appropriate information, and use the Save icon on the toolbar to commit the record to the database.

Nested outline (Remember, to insert into the outline window, there must be a drilldown link between the outline window and a form window based on the same class.) First, the user clicks the Insert button, and Express inserts a blank line. The user then double-clicks in the blank line, invoking the form window, enters the appropriate information, and closes the window. The new information is passed back to the outline field. Users will use the same process to edit data in an outline field—instead of clicking the Insert button, users can simply double-click in the record they wish to edit. Use the Save icon on the toolbar to commit changes to the database.

Creating Multiple Links to the Same Window

The Adaptive Command Interface property in the Business Class Window Properties dialog allows you to create windows that can behave differently when invoked by different links. When this property is on, window behavior is determined automatically by Express, based on a combination of business model, window, and link properties. For example, if one link has a Read Status of Read-Only, then the called window will not have any edit commands. If another link has Read/Write status and is modal, the called window will have edit commands and an OK button. If the business class on which the called window is based is on the “many” end of an association in the business model, then Express will give the window a multiple record command set (either edit or view, depending on the Read Status). The following table summarizes the effect the properties have on a window using the Adaptive Command Interface property:

Setting...	Results in...
Read-Only	View commands
Read/Write	Edit commands
Modal	Button interface
Modeless	Menu/Toolbar interface
Window based on business class at the “many” end of an association	Multiple record commands
Window base on business class at the “one” end of an association	Single record commands
Lookup link	Search and/or View commands

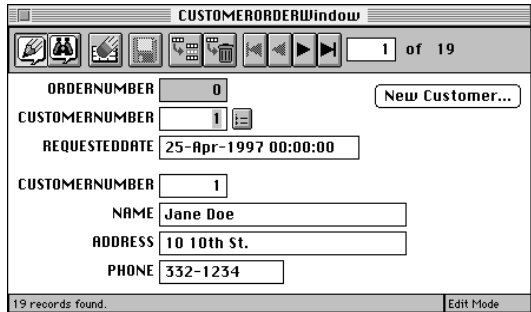
If you set the Adaptive Command Interface property for a window, that property will be in effect for every link coming into the window. Also, Express uses the same field layout for each invocation of a window. Note that if you do not like the choices that Express makes for you, you can customize the generated link method. See *Customizing Express Applications* for more information.

Example Flexible Window

The following example describes a scenario using the Adaptive Command Interface property.

Imagine a CustomerOrder window like the one in the figure below.

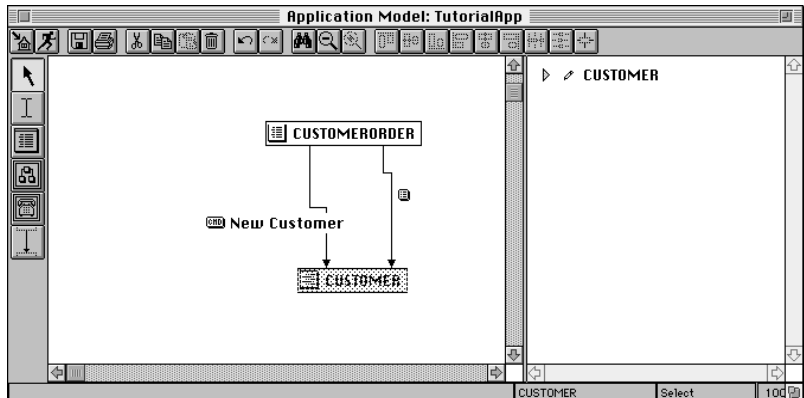
Figure 6-19 CustomerOrder Window



To help a user take orders, it is useful to have a lookup window to the Customer table to retrieve customer numbers. The lookup window needs only scrolling capabilities and OK and Cancel buttons. Another useful feature is to allow a user to enter new customer information. Of course, this information cannot be entered into the order window, so another view into the customer window is required. This version needs to be editable.

You can implement these features by drawing two links to the same Customer window, as shown in the figure below.

Figure 6-20 Multiple Links to Customer Window



You must select the Adaptive Command Interface property in the Customer Properties dialog. By doing so, you tell Express to determine the appropriate commands and interfaces for both versions of the Customer window.

In addition, the Link Properties dialogs for the lookup and command links should look like Figure 6-21 and Figure 6-22, respectively:

Figure 6-21 Link Properties Dialog for Lookup Link

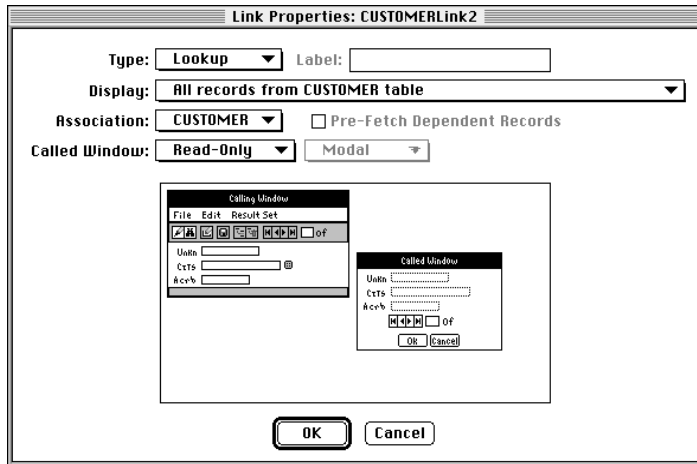
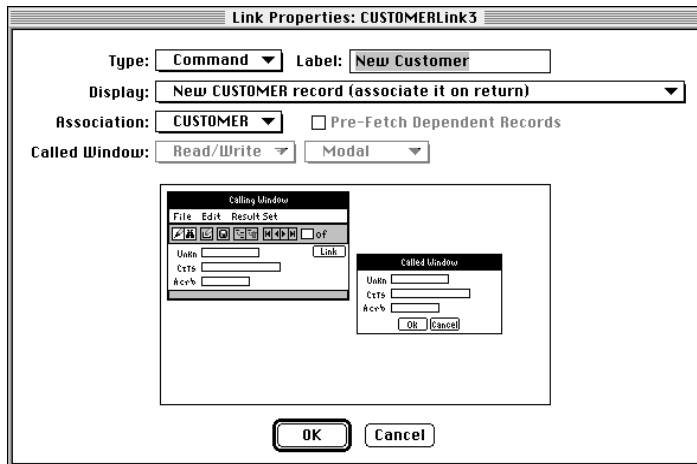


Figure 6-22 Link Properties Dialog for Command Link

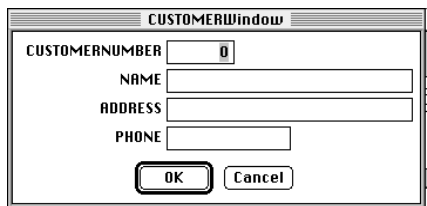


When you run this application, the two different invocations of the Customer window will look like the following

Figure 6-23 Lookup Customer Window



Figure 6-24 Edit Customer Window



Including an Existing Application Model in the Current Application Model

Express allows you to include an existing application model in your current application model by using the Included Application Model tool. Using included applications is a good way to build application “modules” that can be reused in different applications.

► **To include an existing application model**

1. Click the New Included Application Model icon.

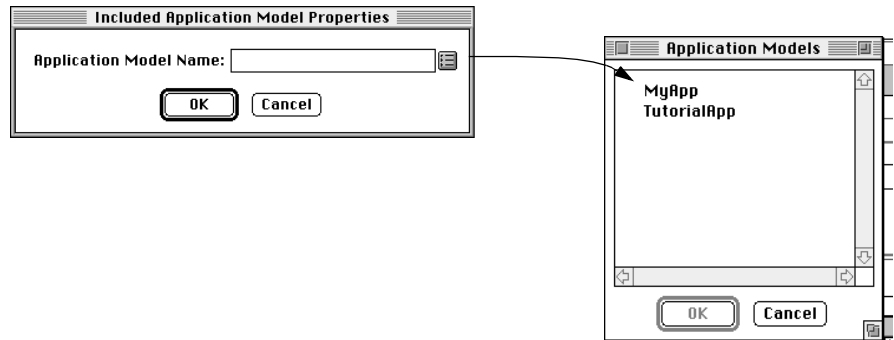


The cursor changes to an up arrow.

2. Click in the drawing area.

The New Included Application Model dialog appears.

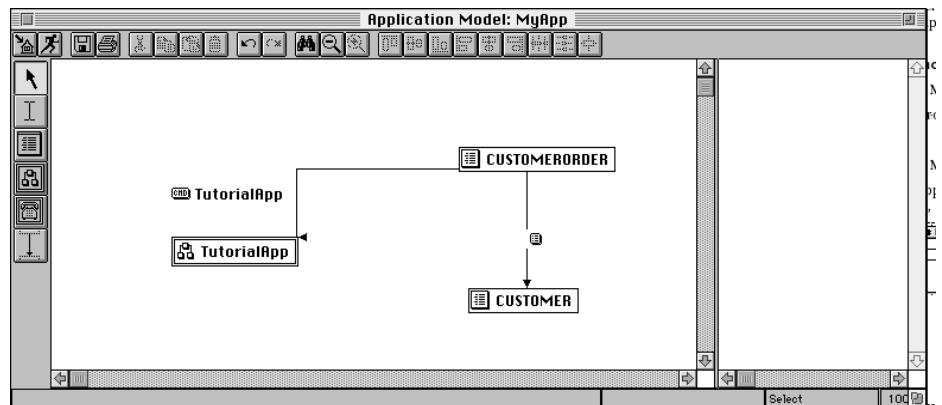
3. Enter the name of an existing application, or click the browser button and select an existing model from the dialog, as shown below.



4. Click OK.

The included application appears in the Application Model Workshop similar to a window icon, but with a double outline, as shown below.

Figure 6-25 Included Application in Application Model



NOTE You can also include an application by opening the Repository Workshop and dragging and dropping an existing application model into the Application Model Workshop.

You can draw links to the included application icon just as you would to any other window. When the included application is called, its start window is invoked. Once the start window is invoked, the application can be navigated as if it had been called directly.

Supplier Plans

By default, Express includes the included application model as a supplier to the current application model. If you delete the included application, Express will remove the application as a supplier to the application model. However, you must manually remove the generated project from the list of supplier plans to the generated project of the current application model.

Using Callouts

An application built using Express can call, directly from its generated code, to user-defined methods, projects, and deployed applications. The New Callout command displays a dialog that lets you choose the following options:

- Custom (for user-defined methods)
- Call Project
- Launch Applet

Callouts to User-Defined Methods

You can write a method call to another window that was created outside of Express, or create a simple message dialog to display information unrelated to the data handled by your application. You use the Callout Tool to create a callout to one of these external objects.

► To create a callout to a custom method

1. Click the New Callout tool or choose the Component > New > Callout command.



The cursor turns into an up arrow.

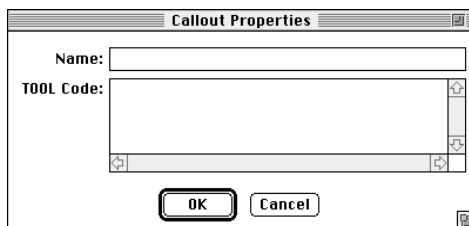
2. Click in drawing area.

The New Callout dialog appears.



3. Click the Custom radio button and then click OK.

The Callout Properties dialog appears.



4. Enter the name of the callout.

This is the name that will appear on the callout icon.

5. Enter the appropriate code in the TOOL Code area.

For example:

```
window.messageDialog('Hello World');
```

6. Click OK.

A Callout icon with the appropriate label appears in the Application Model Workshop.

Callouts to iPlanet UDS Projects

You can use a callout to invoke another iPlanet UDS project. When you use the Call Project option and specify a project, Express automatically includes the project as a supplier to the project that contains the callout, and inserts the appropriate code for the start class and method. Note that in this case the image repository will include project definitions for both applications.

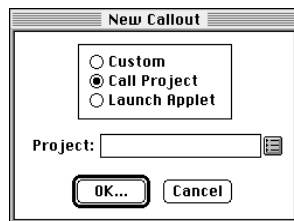
► **To create a callout to a project**

1. Click the New Callout tool or choose the Component > New > Callout command.



2. Click in drawing area.

The New Callout dialog appears.

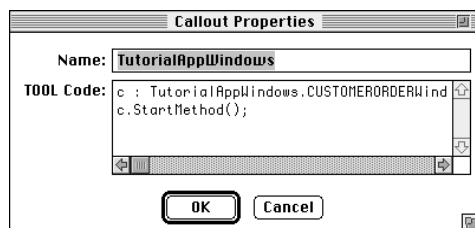


3. Click the Call Project radio button.
4. In the Project field, enter the name of the project.

Alternatively, click the browse button and select a project from the list of projects in the Projects dialog.

5. Click OK.

The Callout Properties dialog appears:



6. Check the launching code for errors, and Click OK.

Note that you can also drag a project from the Repository Workshop and drop it into the Application Model Workshop. Dragging and dropping a project will automatically create a callout to that project.

Callout to an Applet

An applet is a client application that is started as part of another application. In design and function, an applet is actually a small independent application. Express allows you to create a callout to any application that is available on the client node. For information about defining applets, see the *iPlanet UDS Programming Guide*. For information about deploying applications to client nodes, see the *iPlanet UDS System Management Guide*.

► To create a callout to an applet

1. Click the New Callout tool or choose the Component > New > Callout command.



2. Click in the drawing area.

The New Callout dialog appears.

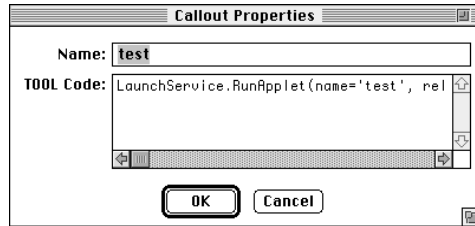


3. Click the Launch Applet radio button.
4. In the Applet field, enter the name of the applet.

Alternatively, click the browse button and select an applet from the list of applets in the Applets dialog.

5. Click OK.

The Callout Properties dialog appears



6. Check the launching code for errors, and Click OK.

Linking a Callout

You create a link to a callout the same way you create a link to any other kind of window.

► To create a link to a callout

1. Select the New Link tool.



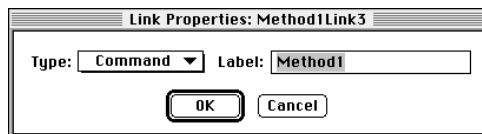
2. Place the cursor in the “calling” window, hold down the mouse button, drag the cursor into the callout, and release the mouse button.

Once you create the link, you will set its properties to determine how the callout code is invoked. Because there is never a relationship between the callout and any calling window (in other words, the link is never based on an association from the business model), you must use a command link to a callout. Callouts can never be nested with the calling window.

► **To set the callout link's properties**

1. Double-click on the link.

The Link Properties dialog appears, as shown below.



Notice that the only property you can change is the Label. Command is the only link type available for callouts.

2. (Optional) Edit the Label text.
3. Click OK.

Callouts to User-Defined Methods and Generated TOOL Code

After you enter the TOOL code in your callout and link to the calling window, Express creates a method that contains your code in the base window class of the calling window. The name of this method is the name of the callout appended with an internal name for the link, appended with Start. For example, if CustomerOrder links to the Method1 callout, then Express creates a method called Method1Link1Start in CustomerOrderBaseWindow. (Link1 is used for the first link to the callout, Link2 for the second link, and so on.) See *Customizing Express Applications* for more information about generated methods.

The generated Start method has access to all the variables, methods, attributes, and so on, of the calling window.

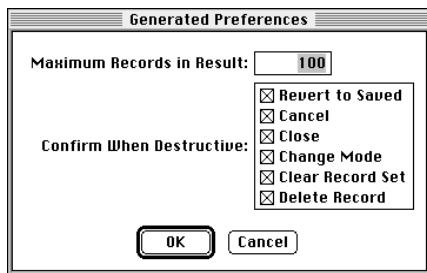
Setting Default Behavior for Generated Windows

You can specify how the generated windows you define in your application model will handle confirmation of potentially destructive operations, such as deleting records. You can also specify the maximum records to retrieve in a query.

➤ **To set generated preferences**

1. Choose the File > Generated Preferences command.

The Generated Preferences dialog appears, as shown below.



2. Enter the maximum number of rows to retrieve.
3. Check the operations you wish to receive a confirmation dialog.
4. Click OK.

The following table describes the dialog properties:

Property	Description
Maximum Records in Result	Specifies how many records can be returned in the result set.
Confirm When Destructive	Determines whether the user is prompted with a dialog to confirm any of the following operations: Revert to Saved Cancel Close Change Mode Clear Record Delete Record

Client Code Generation

Express automatically generates TOOL classes from an application model. Prior to generating your code, you can validate the model to check its consistency. If you have customized classes in the ExpressWindows project, you need to set a custom prefix prior to code generation. You can also force a complete compile to regenerate imported TOOL classes. Each of these topics is described below.

Validating the Model Prior to Generation

The Validate Model command allows you to verify the completeness and consistency of the application model before generating the client code.

For an application model to be valid, the start window for the application model must be set to a valid name.

If there are any errors in your model, these will be displayed in the Errors Found During Model Validation window. You must fix these errors before you can give a Generate Client Code, Force Complete Generation, or Run Application command.

Note that the Generate Client Code, Force Complete Generation, and Run Application commands automatically execute the Validate Model command. If the model is invalid, you must fix the errors before these commands can generate the code or run the application.

► To validate a model

1. Choose the File > Validate Model command.
2. If there are any errors, these are displayed in the Errors Found During Model Validation window; potential errors are described in the following table.

Potential Errors Found During Model Validation

Error Message	Explanation
A valid value for the start window in application model <i>name</i> is required.	The start window for the application must be specified. Use the File > Properties... command to open the Application Model Properties dialog, and set the start window for the application.

Error Message	Explanation
Association <i>name</i> must have to-one or optional multiplicity for link <i>name</i> to be based on it since <i>name</i> is a lookup link.	The specified lookup link is based on an association that does not have to-one or optional multiplicity. You must change the association multiplicity in the Business Model Workshop, or change the type of link you are using.
Because attribute <i>name</i> was deleted from the business class <i>name</i> , Express has deleted the display attribute <i>name</i> from the window <i>name</i> .	An attribute that is currently displayed on the window has been deleted from the business class on which the window is based. Express will delete the attribute from the window.
Could not find a class named <i>name</i> in any supplier business model for application model <i>name</i> .	The business class on which the window is based has been deleted from the business model. You must either recreate the business class in the business model, or base the window on a different business class.
Because association <i>name</i> was deleted from the business class <i>name</i> , Express has deleted the displayed attributes of the business class to which association <i>name</i> points from the window <i>name</i> .	The association on which a link is based has been deleted from the business model. Either base the association on a different link, or recreate the association in the business model.
Name ambiguity: class <i>name</i> is defined in two supplier business models, <i>business model name1</i> and <i>business model name2</i> . One of these suppliers must be removed from application model <i>name</i> .	The same class name is used in two business models, both of which are suppliers to the application model. Change the name of one of the business classes and regenerate.
Because there is no supplier class named <i>name</i> , cannot find association <i>name</i> for link <i>name</i> .	The from-class in an association has been deleted, therefore the association does not exist. You must either base the link on a different association, or recreate the deleted business class and redraw the association. You can also base the window class on a different business class.
Because business class <i>name</i> is no longer a superclass of business class <i>name</i> , attribute <i>name</i> is no longer an inherited attribute of business class <i>name</i> . Express has deleted the displayed attribute <i>name</i> from the window <i>name</i> .	The window currently displays an inherited attribute from a superclass whose generalization has been deleted. To display the attribute, you must redraw the generalization.

Error Message	Explanation
<p>Because business class <i>name</i> is no longer a superclass of business class <i>name</i>, association <i>name</i> is no longer an inherited association of the business class <i>name</i>. Express has deleted the displayed attributes of the business class to which association <i>name</i> points from the window <i>name</i>.</p>	<p>The window, which is based on a subclass, currently displays inherited attributes from the business class to which the superclass's association points. To display the attribute, you must redraw the association.</p>
<p>Aggregation component class window <i>name</i> has an incoming command link <i>link_name</i> and its command set is <i>set</i>. An aggregation component class window which has an incoming command link should not have a command set of All or Search because the search will never select any data.</p>	<p>The association on which a command link is based is an aggregate, and the command set for the linked window is All or Search. A command set of All or Search results in a Search button on the generated window. The Search button will never return any data.</p> <p>Note: If you have an Express Release 1 model that includes this scenario, you will receive the warning message when you import your model.</p>

Setting a Custom Prefix for Generated Classes

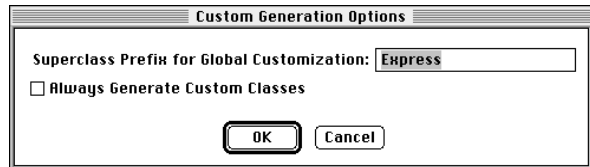
You can customize the `ExpressArrayWindow`, `ExpressFormWindow`, and `ExpressOutlineWindow` classes from the `ExpressWindows` project to alter the appearance of your generated windows. `ExpressWindows` is a supplier plan to every application model.

You first create a new project that contains subclasses of the three `ExpressWindows` classes mentioned above—you must name the subclasses *yourPrefixArrayWindow*, *yourPrefixFormWindow*, and *yourPrefixOutlineWindow*. Note that all the classes must have the same prefix. Make your modifications to the subclasses and include the project as a supplier to your application model. Before you generate code, you must use the Custom Generation Options command under the File menu and enter the prefix (*yourPrefix*) in the Custom Generation Options dialog. See *Customizing Express Applications* for detailed instructions.

► **To enter a custom generation prefix**

1. Choose the File > Custom Generation Options command.

The Custom Generation Options dialog appears, as shown below. If you have not previously changed the prefix, the prefix will be “Express” by default.



2. Enter the same prefix you used in the name of your custom classes.



3. Click OK.

Express searches the supplier projects and looks for classes with the prefix *yourPrefix*. All generated window classes will be subclasses of *yourPrefixArrayWindow*, *yourPrefixFormWindow*, or *yourPrefixOutlineWindow*.

Forcing Complete Generation

If you want to guarantee that all classes in a model will be regenerated, choose the File < Force Complete Generation command. After you import TOOL projects that contain generated classes into your repository, you should use the Force Complete Generation command to regenerate the entire model. The imported TOOL projects may contain previously generated code that is not up-to-date with the current version of Express. Forcing complete generation will bring the imported project back up to date. Note that when you execute this command from the Application Model Workshop, the business models that are suppliers to the current application model will also be regenerated.

Generating Client Code

After you have specified your application model, the next step is to test run the application. Before running the application, you must generate the client code.

For each window class, Express generates a read-only class. For outline windows, an additional read-only class is generated. Using the CUSTOMER window class (set in outline field format) as an example, the generated client classes are:

Generated Class	Superclass	Description
CUSTOMERWindow	ExpressOutlineWindow Other possible superclasses are ExpressFormWindow or ExpressArrayWindow	Contains the generated window. Includes information about the appearance and behavior of the window in the application. <i>A <code>appl_windowWindow</code> class can be a subclass of three different superclasses. Which superclass the generated window class is a subclass of depends on whether you have set the Layout of Fields property to Form, Array, or Outline, respectively.</i>
CUSTOMERNode	DisplayNode	Generated for Outline windows only. Displays data in an outline field.

Customizable subclasses To customize these classes, you must create a customizable subclass of these classes. You can do this with one of the following methods:

- Invoke the Customization Manager on an individual class
This renames the generated class, inserting “Base” into the name, and creates a new, customizable class, which is named with the original class’s name. For example, the CUSTOMERWindow (read-only) becomes CUSTOMERBaseWindow; the new customizable, leaf-level class is then named CUSTOMERWindow.
- Create a full hierarchy (Base and leaf-level classes) for every window class.
You do this by setting the Always Generate Custom Classes option on the File > Custom Generation Options command.

When you use this option, note:

- Turning it on turns it on for all the window classes in the model; it can then only be turned off by deleting all the customizations on each window class using the Customization Manager. If you delete the individual classes you will get errors.
- In other words, you create customizable classes for all window classes in the model with one step, but you must remove individual customizable window classes separately.
- Turning it off will only affect new window classes created from the time you turned it off, resulting in some Base classes having “Base” in their names and some not.

For complete information on creating customizable classes and using the Customization Manager, see *Customizing Express Applications*.

► To generate client code

1. Choose the File > Generate Client Code command.

Express creates a project called *application_modelWindows*, which contains all the iPlanet UDS classes generated from the application model. Refer to *Customizing Express Applications*, for information about the generated classes.

Note that when you generate the client code, Express automatically regenerates the server code if you have made any changes to the business model.

Testing an Application

To test run a Express application, select the Run icon or choose the File > Run Application command. When you test run an application in Express, iPlanet UDS partitions the application so that as much of it as possible will run on your client machine, and any service objects that the client node cannot support are assigned to a suitable node in your development environment. This allows you to test the application independently of its deployment environments. For information about testing an application in a distributed environment, see [Chapter 7, “Partitioning and Deployment.”](#)

When you run an application from within the Application Model Workshop, Express automatically checks to see that all code generation—client and server—is up to date. If the code is not current with models, Express regenerates the appropriate code. If you run from the Repository Workshop or the Project Workshop, Express will not regenerate the code.

Setting Workshop Preferences

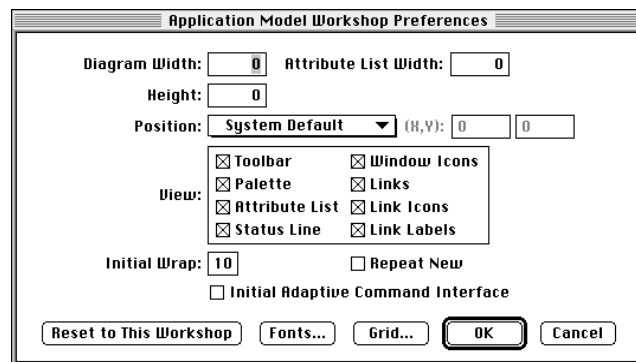
You can customize certain visual and behavioral aspects of the Application Model Workshop to suit your development style. Once you set your preferences, the Application Model Workshop will reflect them each time you open the workshop.

You set Application Model Workshop preferences in the Application Model Workshop Preferences dialog.

► To set Application Model Workshop Preferences

1. Choose the File > Workshop Preferences... command.

The Application Model Workshop Preference dialog appears, as shown below.



Each preference option is briefly described in the following table, and explained in more detail in the sections that follow.

Preference	Description
Diagram Width	Sets the default width of the drawing area.
Height	Sets the default height of the drawing area.
Attribute List Width	Sets the default width of the class list.
Position	Sets the initial position of the Application Model Workshop window in relation to other windows, or the screen.
X,Y	Sets the precise positioning a window should use to place itself, according to the Position setting.
Repeat New	Toggles the Repeat New command in the Component menu to on by default.
View	Specifies which diagram components to view by default.

Preference	Description
Initial Wrap	Specifies the number of attributes to display in a business class before adding scroll bars to the business class.
Initial Adaptive Command Interface	Sets the Adaptive Command Interface property in the Business Class Window Properties dialog to on by default.
Reset to This Workshop	Sets all preferences according to the current workshop.
Font	Sets the fonts used in the Application Model Workshop.

Workshop Size and Position

There are several ways you can set the initial size and position of the Application Model Workshop: capturing the settings of the current workshop, explicitly setting the dimensions of the drawing area and the class list, and specifying the X and Y coordinates of the workshop relative to a specified position.

Capturing Current Workshop Settings

The simplest way to set the initial size and position of the workshop is to do so visually. You can set the window size by using the window's resize handles, and the window position by dragging the window into position. When the window reflects the size and position you desire, choose the File > Workshop Preferences command and click the Reset to This Workshop button in the dialog. This button sets the values of the Position preference to Screen Relative and the values of the X, Y coordinates to the current values of the workshop on the screen.

Note that the Reset to This Workshop button also changes your View preferences to the current settings on the View menu and the font preferences to the current setting in the Application Model Workshop.

Explicitly Setting Window Dimensions

You can control the size of the workshop by specifying the size of the drawing area and the Attribute List area:

Preference	How to specify its value
Diagram Width	Enter an integer that specifies the width of the Application Model Workshop drawing area in mils.

Preference	How to specify its value
Height	Enter an integer that specifies the weight of the Application Model Workshop drawing area in mils.
Attribute List Width	Enter an integer that specifies the width of the attribute list in mils.

Specifying Workshop Position

By setting the Position property, you determine the initial placement of the Application Model Workshop window when the window first runs. You set the window's initial position either in relation to the screen or an application's primary window, or absolutely.

Relative initial window position To set the Application Model Workshop window's position relative to either the screen or to the primary window, you use the Position setting in conjunction with the X and Y specifications. The X setting determines (in *mils*—thousandths of an inch) how far to horizontally offset the window relative to the screen or a primary window. The Y value represents the vertical offset. The primary window is normally the window that opened the current window—in this case is the Repository Workshop or the Business Model Workshop.

Absolute window position To set a window's position absolutely, you choose a position setting of Screen Centered or Primary Centered. These absolute settings ignore the X and Y settings, if any.

The following table summarizes the window position settings.

Position Setting	Definition
System Default	The default position used by the particular window system on which the window is being displayed.
Screen Centered	Centers the window relative to the screen.
Primary Centered	Centers the window relative to the primary window.
Screen Relative	Positions the window relative to the screen, using the X and Y settings to offset the window from the screen.
Primary Relative	Positions the window relative to the primary window, using the X and Y settings to offset the window from the primary window.

Viewing Preferences

You can specify the default settings for menu items on the View menu in the Application Model Workshop. You can make each of the following components visible or invisible:

- Attribute List
- Links
- Link Icons
- Link Labels
- Palette
- Status Line
- Toolbar
- Window Icons

Click the toggle of the components you wish to view; click again to remove from view. You can change this selection on the fly using the View menu.

Drawing Preferences

The Repeat New command in the Component menu allows you to create multiple copies of an application model component without having to reselect its drawing tool. To turn this command on by default, select the Repeat New toggle in the Application Model Preferences dialog.

Business Class Window Dialog Preferences

The following properties specify default settings in the Business Class Window Properties dialog.

Initial Wrap The Initial Wrap property specifies the default number of fields to place either down or across a window before wrapping to the next column or row. You can change this setting on the Business Class Window Properties dialog. This setting affects only the properties of new windows, not existing ones.

Initial Adaptive Command Interface The Initial Adaptive Command Interface property sets the Adaptive Command Interface property for a business class window to on. You can change this setting on the Business Class Window Properties dialog. This setting affects only the properties of new windows, not existing ones. For information about the Adaptive Command Interface property, see [“Creating Flexible Windows” on page 179](#).

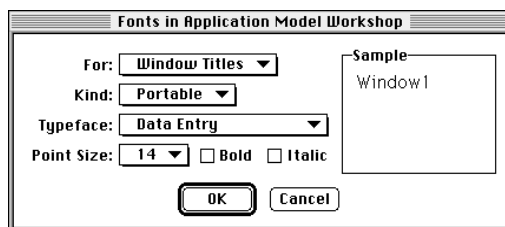
Fonts Preferences

You can set the font to be used for components in an application model. You can individually set font characteristics for window titles, link labels, and the attribute list.

► To set the font for the Application Model Workshop

1. Choose the File > Workshop Preferences... command.
2. Click the Fonts button in the Application Model Preferences dialog.

The Font dialog appears, as shown below.



3. Select the component you wish to configure from the For drop list.
4. Choose the type of font, System or Portable, from the Kind drop list.
5. Select the typeface, size, and style.

The sample illustrates how your font selection will look.
6. Repeat the process for each application model component you wish to specify.
7. Click the OK button to confirm the font changes.

Command Summary

File Menu

Command	Description
Properties...	Opens the Application Model Properties dialog.
Supplier Plans...	Specifies supplier plans for the application model.
Fonts...	Controls the fonts used in the current application model. These settings override the default fonts set by the Workshop Preferences command.
Generated Preferences...	Opens a dialog allowing you to specify the maximum number of records to return in a result set, and whether to display a confirmation dialog for destructive commands.
Validate Model	Verifies model consistency.
Generate Client Code	Generates iPlanet UDS window classes based on the information in the application model.
Custom Generation Options...	Opens the Custom Generation Options dialog, which displays the prefix to use for the generated classes.
Force Complete Generation	Specifies that all classes be regenerated.
Run Application	Runs the application specified by the application model.
Save All	Saves all changes made to the workspace.
Workshops Open...	Opens the Workshops Open dialog, which displays a list of all the workshops that are currently open.
Workshop Preferences...	Invokes a dialog that allows you to set a variety of characteristics of the Application Model Workshop, such as the size of the drawing area and the components to display.
Print...	Prints the current diagram.
Page Setup...	Opens a dialog in which you control page and printer settings.
Cancel	Cancels any changes made since the last save.
Close	Saves and closes the workshop.

Edit Menu

Command	Description
Undo	Undoes the last action performed. You can repeat this command as many times as you wish, until you reach the first action you performed in the workshop, or the last save. This command also undoes any information you added to property dialogs—the entire property dialog is emptied as one action.
Redo	Re-does the last action you reversed using the Undo tool.
Cut	Cuts the selected component to the clipboard.
Copy	Copies the selected component to the clipboard.
Paste	Pastes a cut or copied component from the clipboard.
Clear	Deletes the selected components.
Delete	Deletes the selected components.
Select All	Selects all the components of the application model.
Find...	Displays a dialog in which you enter the name of a window. Use this command to locate and display a window not currently visible in a large application model.

Component Menu

Command	Description
New	Provides a slide-off menu for creating windows, included application models, callouts, and links.
Repeat New	Turns on or off Repeat mode for creation of components.
Properties...	Invokes the Properties dialog for the selected component.
Customize...	Invokes the Customization Manager on the currently selected window.
Class Workshop	Invokes the Class Workshop for the class generated from the selected business class window.
Attribute Properties	Invokes a dialog allowing you to modify the status of the selected attribute in the Attribute List.

View Menu

Command	Description
Toolbar	Toggles on and off the display of the toolbar.
Palette	Toggles on and off the display of the palette.
Attribute List	Toggles on and off the display of the Attribute List.
Status Line	Toggles on and off the display of the status line.
Window Icons	Toggles on and off the display of the window class icons.
Links	Toggles on and off the display of the links.
Link Icons	Toggles on and off the display of the link icons.
Link Labels	Toggles on and off the display of the link labels.
Grid...	Displays the Grid dialog, where you can choose to display a grid, set grid spacing, and turn on the Snap to Grid property.
Zoom In	Doubles the size of the application model.
Zoom Out	Reduces the size of the application model by 50%.
Zoom...	Displays the Zoom dialog, in which you specify the percentage view to zoom to. Choose a preset value from the radio list, or enter a custom value.

Arrange Menu

Command	Description
Align Windows to	Aligns the selected window classes to either the top, middle, bottom component, left, center, or right component.
Distribute Windows	Distributes selected classes along a horizontal or vertical axis.
Distribute Connections	Evenly spaces the endpoints of links connected to the selected windows.

Partitioning and Deployment

This chapter provides information about how to partition, test, and deploy the applications you create using Express. Topics covered include:

- service objects in Express applications
- default partitioning for Express applications
- modifying the configuration
- testing the application
- deploying the application

Partitioning Express Applications

After generating your code for your Express application and making any desired customizations, you should use the Partition Workshop to:

1. Test the application in a distributed environment.
2. Partition the final application for deployment and make the appropriate application distributions.

The following sections provide background information about the projects in the Express application, the service objects in Express applications, and the default partitioning scheme iPlanet UDS provides for Express applications.

About Express Projects and Service Objects

As described in *Customizing Express Applications*, when you generate the code for your Express application, two or more iPlanet UDS projects are created:

Project Name	Description
<i>application_model</i> Windows	The main project for the application.
<i>business_model</i> Services	A supplier project for the “Windows” project. There is one “Services” project for each business model that is a supplier to the application model.

Partitioning the “Windows” project The “Windows” project generated from the application model is the *main project* for the application. The “Services” project generated from the business model functions as a *supplier project* to the main project. Therefore, when it is time to test your application in a distributed environment or partition the application for deployment, you must partition the “Windows” project.

If your application model is based on more than one business model, there will be one Services project for each business model. All Services projects are supplier projects to the Windows project.

When you partition your Express application using the Partition Workshop, iPlanet UDS creates a default configuration for the application based on the application’s service objects. Both the “Windows” and “Services” projects contain service objects that affect this default configuration.

Briefly, the service objects in your Express applications are:

Service Object	Partition	Description
TheBroker	Client	Provides services needed by the user interface. You should never move or modify this service object.
<i>service_name</i> Service	Server	The application contains one Service service object for each Express service defined in the Business Model Workshop. You can change the replication of this service object to provide load balancing and/or failover.

Service Object	Partition	Description
<i>service_name</i> DBService	Server	The application contains one DBService service object for each Service service object. The DBService service object will always be in the same partition as the Service service object with which it is associated.

The following two sections provide further information about these service objects.

TheBroker Service Object

The TheBroker service object is defined in the Windows project for your Express application and provides services needed by the user interface. You should never move this service object or modify its definition.

Service and DBService Service Objects

In the Business Model Workshop, you create the Express services that manage the business classes in your application. These Express services define the Service and DBService service objects for your application.

By default, there is only one Express service in the application, which has the same name as the business model. For example, the TutorialApp application has one Express service called “Tutorial.” The Tutorial service creates two service objects:

- TutorialService
- TutorialDBService

However, you may wish to create more than one Express service. The ability to create additional services for your Express application provides you with more flexibility for partitioning the application.

For example, creating more than one Express service allows you to distribute processing across different nodes in the deployment environment by assigning the corresponding service objects to different nodes. In addition, creating more than one Express service allows you to provide different replication options (load balancing or failover) for the corresponding service objects.

Another reason you may wish to create additional Express services is if you want your application to interact with more than one database. Each service is bound to a particular database session, so you must create a separate service for each database you wish to access. You can then assign the corresponding service objects to the nodes that have the appropriate resources.

Creating new services In the Business Model Workshop, the Component > New Service command (described in [“Creating New Express Services” on page 129](#)) allows you to create additional Express services. Each time you create a new service, Express adds two new service objects to your application—a Service service object and a DBService service object. The Service service object provides the processing for all interactions between the client and the database. The DBService service object provides the individual database session needed by the Service service object to access the database.

The following two sections provide further information about Service and DBService service objects. [“Relationship between Service and DBService Service Objects” on page 227](#) provides information about the special relationship between these two kinds of service objects.

Service Service Objects

There is one Service service object for each Express service in the business model. The Service service objects are defined in the Services project for your Express application. Each Service service object is based on a *service_name*ServiceMgr class in the Services project and has the following name:

*service_name*Service

The initial properties of a Service service object are determined by the properties you set for the corresponding Express service in the Business Model Workshop. For example, if the dialog duration for the Express service is message duration, the dialog duration for the corresponding Service service object is message duration. See [“Express Services” on page 125](#) for information on setting properties for an Express service.

By default, replication is turned off for all the Service service objects. However, you can turn on replication for an individual Service service object by modifying the service object’s definition in the Partition Workshop. See [“Modifying the Configuration” on page 232](#).

DBService Service Objects

There is one DBService service object for each Express service in the business model. The DBService service objects are defined in the Services project for your Express application. Each DBService service object is based on the DBSession class in the GenericDBMS library and has the following name:

*service_name***DBService**

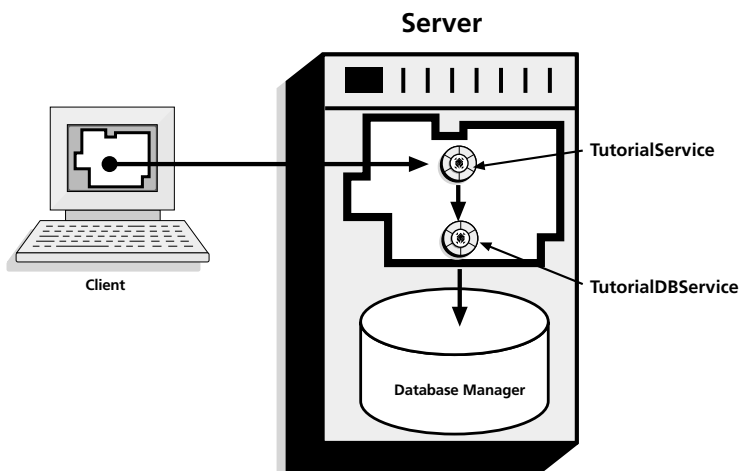
Each DBService service object represents a database session for a particular database resource manager. The specific database session information for the service object was determined by the properties you set for the corresponding Express service in the Business Model Workshop. For example, if the resource manager for the Tutorial service is “TestOracle,” the resource name for the TutorialDBService service object is TestOracle. See [“Editing Service Properties” on page 134](#) for information on setting properties for an Express service.

However, you may wish to change the resource manager name, database name, user name, or user password for an individual DBService service object by modifying the service object’s definition in the Partition Workshop. See [“Modifying the Configuration” on page 232](#).

Relationship between Service and DBService Service Objects

Every Service service object has a one-to-one relationship with a DBService service object. The DBService service object provides the database session used by the Service service object to access the database. Both the Service and DBService service objects must always be on the same partition. The following figure illustrates:

Figure 7-1 Relationship between Service and DBService Service Objects



Express uses this architecture for your application because the Service service object provides DBMS access *through* a user-visible DBSession service object (the DBService service object). A partition that includes both an environment-visible service object (the Service service object) and a user-visible service object (the DBService service object) makes the environment-visible service object shared for the application and the user-visible service object private for that partition. The user-visible service object can be accessed indirectly *through* the environment visible service object—neither the client partition nor any other service objects can access it directly. With this architecture, a client can access the database only through the Service service object. The DBService service object is protected from inappropriate access, because no other partitions can access it.

See [“Moving Service Objects” on page 233](#) for information about moving the Service and DBService service objects.

Default Partitioning for Express Applications

When iPlanet UDS partitions an Express application, it assigns all the service objects to logical partitions, one client partition and one or more server partition.

Client partition The client partition contains the user interface for the application, the application startup code, and TheBroker service object. The client partition is assigned to every client node in the environment.

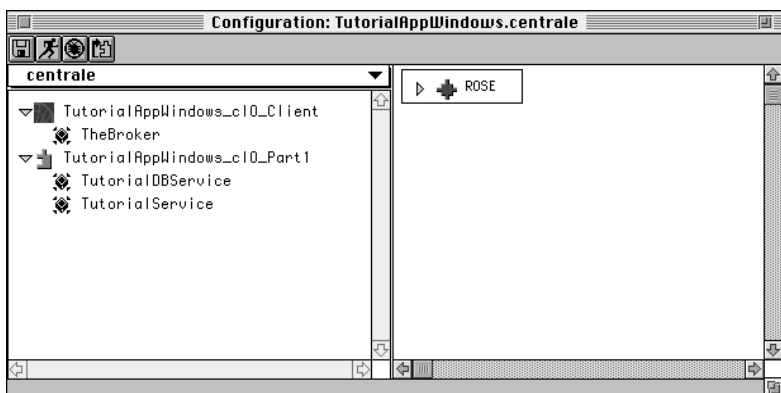
Server partitions The server partitions for an Express application contain the Service and DBService service objects from the Services project. The Partition Workshop assigns all Service and DBService service objects for all the Express services to a single partition unless the external resource managers with which the services are associated are different types or are on different nodes.

When the external resource managers for particular services differ in type and/or location, the Partition Workshop creates separate server partitions for each resource manager type and/or location.

The Partition Workshop then assigns each server partition to the node where the required external resource is installed. If you have previously specified a default node and/or excluded nodes for the configuration, this will be taken into account when the application is partitioned.

The following figure illustrates the default configuration for the TutorialApp application you created in [Chapter 3, “Express Tutorial.”](#)

Figure 7-2 Default Configuration For TutorialApp



Unassigned service objects If you have any DBService service objects in the application that cannot be supported in the current environment (for example, because the appropriate database management system is not present in the environment), both the DBService service object and its corresponding Service service object will be *unassigned*. To run your application in this environment, you must either update the DBService service object definition so it can be assigned to a partition, or you must define the appropriate resource manager in your environment. (See [“Modifying Service Object Definitions”](#) on page 232 for information about updating the DBService service object definition.) You can then assign the partition to the appropriate node.

Testing the Application In a Distributed Environment

If you wish to test a client application in a distributed environment, you use Partition Workshop.

➤ **To test a client application from the Repository Workshop**

1. Select the “Windows” project for your Express application.
2. Choose the Plan > Run > Partition... command or click the Partition Workshop icon.



3. In the Partition Workshop, select the environment from the environment drop list.

➤ **To test a client application from the Partition Workshop**

1. In the environment drop list, select the environment.

Selecting an environment The environment drop list shows all the environments in your environment repository (see *iPlanet UDS System Management Guide* for information about the environment repository and about defining environments). Note that if any environments are currently locked by the Environment Console or the Escript utility, you will not be able to use them.

Before running, you may wish to modify the configuration. See [“Modifying the Configuration” on page 232](#).

Run command To run the application, choose the Run > Run command, or click the Run button on the toolbar. This command works exactly like the Run Distributed command in the Repository and Project Workshops.

Cancel Run command If you are unable to exit the application, you can use the Run > Cancel Run command at any time to cancel execution. This command cancels the client partition for the application. The remote partitions will still continue to run; use the Run > Stop Remote Partitions command to stop remote partitions.

See *A Guide to the iPlanet UDS Workshops* for further information about testing applications in a distributed environment.

Deploying the Application

To deploy an Express application, you must partition the application for each deployment environment in which it will run. After the application is correctly partitioned, you must create a separate distribution for each deployment environment.

► To deploy your Express application

1. In the Repository Workshop, double-click the Windows project for your Express application.
2. In the Project Workshop that opens, choose the File > Configure as > Client command.
3. In the Partition Workshop that opens, select the environment from the environment drop list.

This opens the configuration for the project or creates a default configuration if none already exists.

4. Modify the configuration as desired.
5. When the configuration is complete, use the File > Make Distribution command to make the application distribution.

The application distribution is a representation of the application outside of the repository that is used to install the application in an environment.

6. The system manager must take the application distribution and install it in the environments (see the *iPlanet UDS System Management Guide* for information).
7. If you are deploying in more than one environment, select the next environment, make modifications to the configuration if necessary, and use the File > Make Distribution command to create the distribution. Repeat for any number of environments.

See *A Guide to the iPlanet UDS Workshops* for complete information about partitioning applications and making distributions.

Compatibility level Note that if you have already deployed one version of the application and wish to deploy another, you should raise the compatibility level of the Windows project for your Express application. Raising the project's compatibility level lets you install and run the new release of the application in the same environments where older versions of the application are installed. To raise the compatibility level, open the project in the Project Workshop, and give the File > Properties... command to open the project's properties dialog. See *A Guide to the iPlanet UDS Workshops* for information on setting a project's properties.

Modifying the Configuration

A Guide to the iPlanet UDS Workshops explains in detail how to modify a configuration. This section provides some hints about typical changes you may wish to make to the configuration for an Express application, including:

- modifying service objects
- moving service objects
- replicating partitions
- sharing a business service between applications

Modifying Service Object Definitions

Typically, you will need to change the definitions of the service objects in your configuration. For Service service objects, you may wish to specify that the service object be replicated for load balancing and/or failover. For DBService service objects, you may need to change the external resource name, database name, user name, or user password for the DBMS session.

➤ **To change the service object definition**

1. In the Logical Partitions browser, double-click the service object name to open the Service Object dialog.
2. In the Service Object dialog, update the appropriate properties.

Properties that you are not allowed to change will be read only.

Note that changing the definition of a service object to replicated will cause it to move from the original server partition to a replicated partition.

Moving Service Objects

You can move any Service service object to any compatible partition. For the partition to be compatible, the service objects already in the target partition must meet the following conditions:

- If the service object you want to move is replicated, the service objects in the target partition must be replicated the same way.
- If the service object you want to move is associated with a resource manager, the service objects in the target partition must be associated with the same resource manager, if any.

If you do try to move a service object to an incompatible partition, you will get an error message that explains why it is incompatible.

► To move a service object

1. In the Logical Partition browser, select the Service service object you wish to move.
2. Drag the Service service object on top of the target partition name.

Both the Service service object and the DBService service object associated with that Service service object will automatically move to the target partition. If the partition is incompatible, iPlanet UDS will display an error dialog explaining why the service object cannot be moved.

As described under [“Relationship between Service and DBService Service Objects” on page 227](#), there is a special relationship between the Service and DBService objects defined by a given Express service. Both the Service and DBService service objects must always be on the same partition.

While you can move a Service service object from one partition to another, the Partition Workshop does not allow you to move a DBService service object. And when you move a Service service object, the DBService service object associated with the Service service object automatically moves with the Service service object.

Replicating Partitions

After you have changed the definitions of one or more Service service objects to allow replication, you can replicate the partition or partitions that contain them. A replicated partition is a partition that contains a service object defined as replicated for load balancing or failover. When a logical partition is replicated, you can assign it to any number of nodes in the environment.

► **To assign a logical partition**

1. In the Logical Partitions browser, select the logical partition you wish to assign.
2. Drag the logical partition to the node to which you wish to assign it.

The node must provide the resources necessary to run the particular partition.

Router partition When the service object in a partition is replicated for load balancing, iPlanet UDS automatically creates an extra partition called a router partition. The purpose of a router partition is to route the traffic between the partitions that are load balancing work for the service. Although the router partition is usually assigned to the same node as one of the server partitions that it is managing, it can be on any server node in the environment. You can move it if you wish.

Sharing a Business Server Between Applications

As described in [“Creating a Business Class Window” on page 172](#), you can create more than one application model based on the same business model. In other words, the same business model can be used as a supplier plan to any number of application models. Creating more than one application model from the same business model creates multiple applications based on a single business service definition.

Normally, if you deploy two different applications based on the same business service, iPlanet UDS starts a separate copy of the business server for each application. However, you may wish to share the same business server between multiple applications. You can do this by using a *reference partition*.

Reference partitions If one application that includes the business service has already been deployed, you can create a second application that shares the same business server. In your new application, you must include the business model that defines the iPlanet UDS business service as a supplier plan. Then, in the Partition Workshop, you can create a reference partition for the iPlanet UDS business service in your new application. Instead of containing new service objects, the reference partition points to the existing service objects that were originally deployed as part of the first application.

Reference partitions allow you to create business services that are shared by any number of applications in the current environment. If your deployment environments are connected, you can create business services that are shared across environments. The advantages of using reference partitions include:

- modularity

The ability to share a single service object between multiple applications means that you need to create and manage only that single service. Without reference partitions, you would need to create more than one version of the service.

- efficient use of resources

Only one service object needs to be running in order to provide services needed by multiple applications. Without reference partitions, you would need to run the service object within every application that needs its services.

➤ **To include a shared service in your application**

1. Deploy the first application that includes the business model defining the service that is to be shared.
2. In the application model that needs to access the shared service, include as a *supplier plan* the business model from the deployed application that originally defined the service.
3. In the Partition Workshop, choose the Component > New Reference Partition command to create a reference partition that points to the existing service objects.

The New Reference Partition command in the Partition Workshop creates a new reference partition, and allows you to select the service object in the other application that you wish to access.

➤ **To make a reference partition**

1. In the Logical Partition browser, select the service object name that you wish to access from the reference partition. For Express applications, you should select the Service service object that defines the business service (the DBService service object associated with the Service service object will automatically be included in the partition).
2. Choose the Component > New Reference Partition... command.
The Select Containing Application dialog appears.

3. In the Select Containing Application dialog, select the application that contains the existing service object you wish to reference, and click the OK button.

If you are sharing a service between connected environments, you will need to take a few extra steps.

See *A Guide to the iPlanet UDS Workshops* for complete information on using the Partition Workshop to create a reference partition.

Using Generated Windows

This chapter describes how to use windows generated by Express.

Using Windows

This section describes how to use windows generated by Express. In the discussion for each type of window, assume that the window has been built using the All database command set.

About Result Sets

Windows work with data that is either retrieved directly from the database or passed from another window. This data is called a result set.

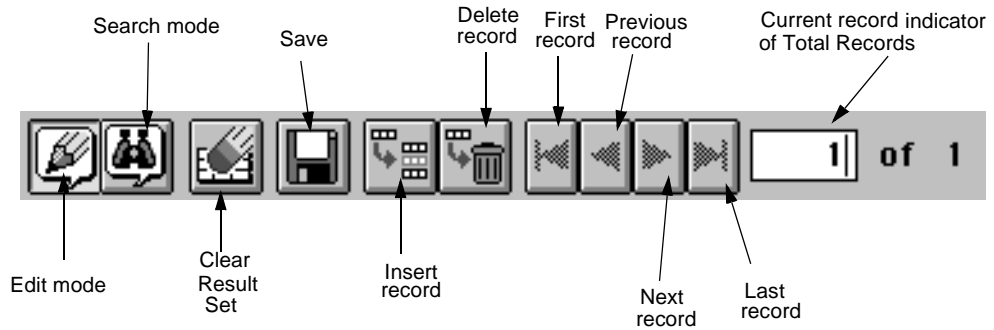
The end user can manipulate the results set using the Edit commands. Changes made to the result set are not returned to the database until the user invokes the Save command.

About Window Modes

Windows that use the All database command set will be able to retrieve and manipulate data from the database. The All database command set includes Search Mode and Edit Mode commands, allowing the user to place the window in Edit mode or Search mode. Edit mode provides all the commands for editing, scrolling, and updating the database. Search mode provides commands for performing query by example.

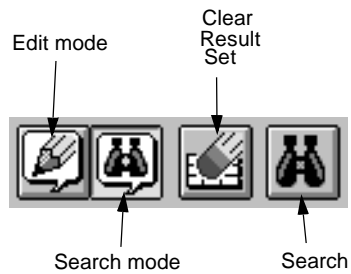
When users first invoke a window using the All command set, it appears in Edit mode, and displays the commands shown in the figure below.

Figure 8-1 Edit Mode Commands



When the end user clicks the Search Mode icon, the toolbar changes to display the Search commands, as shown in the window below.

Figure 8-2 Search Mode Commands



When Express completes a search, the result set is returned to the window in Edit mode.

If a user changes modes after making changes to the result set and has not saved, then window can optionally display a dialog to confirm the change of mode.

Simple Window Types

There are three types of window layout: form, array, and outline.

Form Window

A form window displays one record of the result set at a time. The exact layout of the window depends on the properties you set in the Window Properties dialog. A form window is displayed in the figure below. The current record is the displayed record.

Figure 8-3 Form Window

CUSTOMERORDERWindow

5 of 19

ORDERNUMBER 150

CUSTOMERNUMBER 1

REQUESTEDDATE 28-Mar-1997 16:08:21

19 records found. Edit Mode

Array Window

An array window displays all the records from the result set, along with scroll bars to view records that do not fit in the window. Users can edit each field in the array. The current record is the highlighted record.

Figure 8-4 Array Window

CUSTOMERORDERWindow

1 of 19

ORDERNUMBER	CUSTOMERNUMBER	REQUESTEDDATE
0	1	25-Apr-1997 00:00:00
2	1	25-Apr-1997 00:00:00
26	1	19-Nov-1996 01:00:00
100	33	21-Mar-1997 15:30:50
150	1	28-Mar-1997 16:08:21

19 records found. Edit Mode

Outline Window

An outline window displays all the records from the result set, along with scroll bars to view records that do not fit in the window. Users cannot edit records directly from the outline window. You must create a link to a form or array window to allow them to do so. The current record is the highlighted record.

Figure 8-5 Outline Window

ORDERNUMBER	CUSTOMERNUMBER	REQUESTEDDATE
0	1	25-Apr-1997 00:0
2	1	25-Apr-1997 00:0
26	1	19-Nov-1996 01:0
100	33	21-Mar-1997 15:3
150	1	28-Mar-1997 16:0
152	1	21-Mar-1997 16:3
232	1	25-Nov-1996 01:0
323	40	12-Dec-1996 01:0
543	2	22-Nov-1996 01:0

On Windows95 and NT platforms, outline windows display list views, which let you sort rows and resize columns.

Inserting Records

Form windows When users first invoke a window, it is displayed in edit mode, and may or may not contain data. To insert data, the user must choose the Insert Record command, and then enter the data. To insert the next record, the user must select the Insert Record command, enter the data, and repeat for each new row. Selecting the Save command commits the rows to the database. At any point before saving, the new records can be discarded (using Clear Result Set)—this will discard all the newly created records (not just the last record). The user may also delete newly inserted records.

Array windows To insert data into an array, the user must choose the Insert Record command. If the window is empty, then a blank line appears into which the user can enter the information. If the array window displays data, the blank line appears beneath the last row. Each time you choose Insert Record, a blank line appears to hold the new record. Note that users will be able to insert records into an array window only if the window was specified with the Buttons default interface and the Multiple Record Edit command set.

Outline windows Records cannot be inserted directly into an outline window. When an end users choose the Insert Record command, a blank record appears. You must supply a link to another window in which the user can supply values for the fields in the record. Upon closing that window, the modified record is returned to the results set of the outline window, and appears in the previously blank line.

Note that users can insert records into an outline window only if the window was specified with the Buttons default interface, and the Multiple Record Edit command set.

Deleting Records

Delete behavior is the same for all three window types. To delete the current record, the end user chooses the Delete Record command. The Delete Record command removes the current record from the results set; to remove the record from the database, the end user must select the Save command before closing the window. You can optionally supply a dialog to confirm the save.

After the record is deleted, the next record becomes the current record.

Note that users can delete records from outline and array windows only if the windows were specified with the Buttons default interface, and the Multiple Record Edit command set.

Retrieving Data

To retrieve all rows from the table on which the window is based, the user enters search mode by choosing the Search Mode command, and then selects the Search command. All the rows in the table are returned in the result set.

Form windows display one record at a time, regardless of the size of the result set. Array and outline windows use scroll bars to view records that do not fit in the window.

Query by example To retrieve a record based on specific search criteria, the end user must first select the Search Mode command. When the window is in search mode, the user types the search criteria into the appropriate field or fields, and then clicks the Search command. Only the records that match the search criteria are returned in the results set. You can query on any field in the record.

Searching text fields What you can use for search criteria depends on the datatype of the field used for the search. For text fields, Express searches a match on whatever text you type in, and also supports the percent (%) wildcard. Any other wildcard mechanisms are database dependent.

For example, to find all names starting with J, type `J%` in the Name field and click Search. To find all records matching the string Jones, type `JONES` in the Name field and click Search.

Searching numeric and date time data fields To search for numeric data (including floating point, integer, or money data), or for date time data Express supports the following search syntax:

[[**not**] *operator*] *value*

[**not**] **null**

[**not**] **between** *value* **and** *value*

And for numeric data only:

[**not**] **in** (*value* [*value*...])

The following table explains each of the variables:

This variable	Symbol	Meaning
<i>operator</i>	>=	Greater than or equal to
	<=	Less than or equal to
	=	Equals
	>	Greater than
	<	Less than
	<>, !=, or ^=	Not equal to
<i>value</i>	Any string of characters not containing the characters (space), (,), <, >, =, !, or ^. If the field has a template, the string must follow the template format (see example, below).	

Money data queries You must enter data according to the template format applied to the field. For example, if the money data template is formatted as \$#,##0.0, and you query on all records that have the amount of \$2000, you must enter:

\$2,000

If you forget the comma, Express will display a format error. However, the comma is not necessary for amounts less than \$1,000. To find all orders valued over \$25, enter:

>\$25

Using “between...and” To find all parts with a part number in the range of 700 to 750, enter the following string for your query:

```
between 700 and 750
```

Querying on NULL To check that all orders are non-null, you can query on NULL to see if someone mistakenly entered one. To query on NULL, enter:

```
null
```

Querying by date Both date and time data are always included in date fields, because iPlanet UDS has only the DateTime data type. When a field has a template that specifies both date and time, each record entered on a given day will have unique time data. When a field has a date-only template, all records entered on a given date will have “00:00:00” entered in the database for the time. These facts are illustrated in the table below.

Template	Data entered	Data in database
m/d/yy hh:mm:ss Local	3/27/97 08:35:55 am	03-mar-1997 08:35:55 am
m/d/yy	3/27/97	03-mar-1997 00:00:00

Note that the formats of the DATETIME and DATE templates are determined by the formatting of date and time of the local platform.

To query by date requires taking into account whether or not the field had a template or mask that included time data when the record was created.

Assume the Requested Date field currently has a “m/d/yy” template. Prior to May 20, 1997, the same field had a “m/d/yy hh:mm:ss Local” template.

To query on all records dated Feb. 14, 1997 (created when the “m/d/yy hh:mm:ss Local” template was in effect), type:

```
between 2/24/97 and 2/25/97
```

To query on all records dated May 25, 1997 (created after the “m/d/yy” template came into effect), type:

```
5/25/97
```

Scrolling

To scroll through the results set in a window, use the set of scrolling commands.

A form window displays only one record at a time, so each time the end user selects a scroll command, the window will display a different record.

Array and outline windows display multiple records, highlighting the current record.

Scroll forward The Next Record command scrolls forward one record each time it is selected. The Current Record of Total Records indicator will increment with each record.

Scroll backward The Previous Record command scroll backward one record each time it is selected; this command is available for records other than the first. The Current Record of Total Records indicator will decrement with each record.

Scroll to first record First Record command displays the first record in the results set. The Current Record of Total Records indicator will display 1.

Scroll to last record The Last Record command displays the last record in the results set. The Current Record of Total Records indicator will display the number of the last record, which will match the total number.

Clearing the Result Set

The Clear Result Set command clears a window of all data and empties the underlying results set. This command has no effect on the database, it simply clears the fields and empties the client-side cache of records. For example, if a user has inserted several records and then decides that he or she does not want them, using the Clear Result Set command to discard the new records, along with emptying the existing result set. At this point, if the user wishes to work with existing data, she will have to perform a search. Alternatively, if the user wishes to insert new records, she can do so into the now empty result set.

You can optionally supply a dialog to confirm this operation when the user closes the window.

The behavior of this command is the same for all three window types.

Reverting to Last Saved

The File > Revert to Last Saved command discards any changes made to the result set and displays the previously returned result set.

You can optionally supply a dialog to confirm this operation when the user closes the window.

Saving Changes to the Database

When you edit the result set, which includes modifying, adding, or deleting records, the changes are not saved to the database until you choose the Save command. If you try to exit the window without saving, a confirmation dialog appears allowing you to save your changes at that time. Choose Yes to save the changes, choose No to discard the changes and close the window, or choose Cancel to cancel the close operation.

Using Links

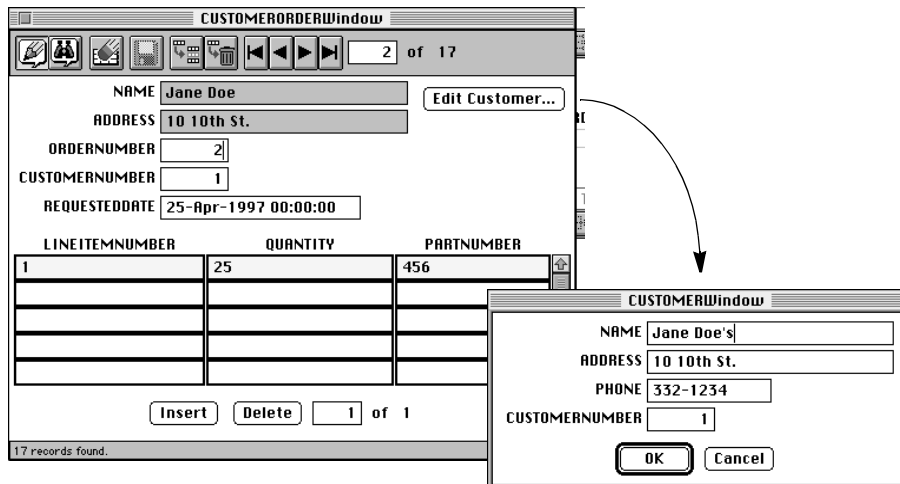
This section describes how to use each of the following links:

- command
- drilldown
- lookup
- nested, including folder

Command Links

When you link two windows with a command link, the calling window always receives a button with which to invoke the called window, as shown in the figure below.

Figure 8-6 Command Link

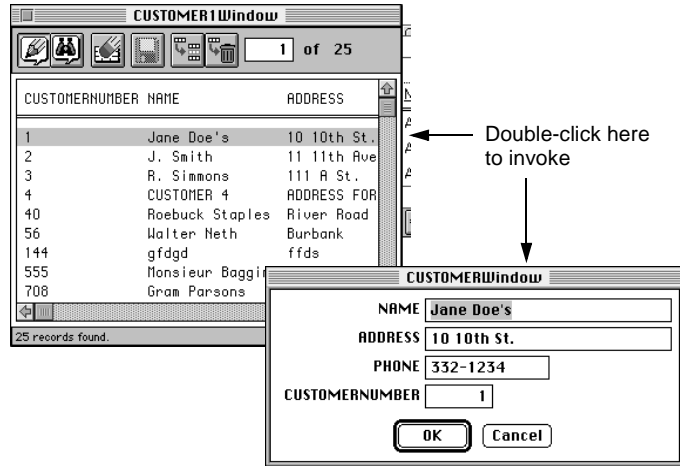


After you insert or retrieve the desired record, close the called window and the record is passed back to the calling window.

Drilldown Links

Drilldown links are available from outline windows. You invoke a drilldown link by double-clicking in an outline record. You can use drilldown links to view detail data, or insert data into an outline.

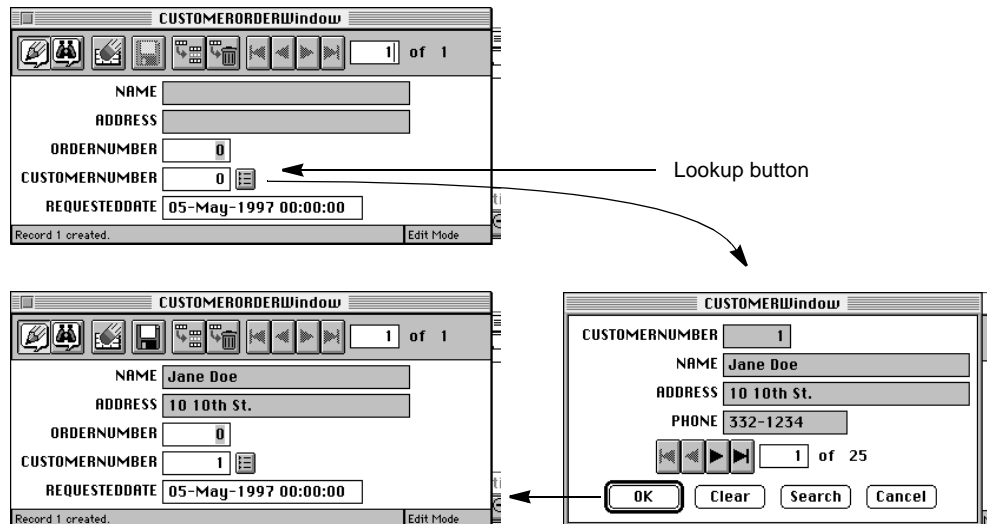
Figure 8-7 Drilldown Link



Lookup Links

Lookup links always call a separate window in which users can search for data. The selected record is always returned to the calling window after the user closes the called window.

Figure 8-8 Lookup Link



Nested Links

You use nested links to make two windows work and appear as one. The figure below is an example of window created using a nested link.

Figure 8-9 Nested Link

The screenshot shows a window titled 'CUSTOMERORDERWindow'. At the top, there is a toolbar with various icons and a status indicator '2 of 17'. Below the toolbar are several form fields:

- NAME: Jane Doe
- ADDRESS: 10 10th St.
- ORDERNUMBER: 2
- CUSTOMERNUMBER: 1
- REQUESTEDDATE: 25-Apr-1997 00:00:00

Below the form fields is a table with the following columns: ORDERNUMBER, LINEITEMNUMBER, QUANTITY, and PARTNUMBER. The first row contains the values 2, 1, 25, and 456. There are three empty rows below it. At the bottom of the window, it says '17 records found.' and 'Edit Mode'.

ORDERNUMBER	LINEITEMNUMBER	QUANTITY	PARTNUMBER
2	1	25	456

Folder links Another type of nested link is a folder link (also called a tab folder link). A folder link also makes two windows work and appear as one, but the nested window looks visually like a folder. You use a folder link to nest multiple windows.

Figure 8-10 Folder Link

The screenshot shows a window titled 'CUSTOMERORDERWindow'. At the top, there is a toolbar with various icons and a status indicator '2 of 17'. Below the toolbar are several form fields:

- NAME: Jane Doe
- ADDRESS: 10 10th St.
- ORDERNUMBER: 2
- CUSTOMERNUMBER: 1
- REQUESTEDDATE: 25-Apr-1997 00:00:00

Below the form fields, there are two tabs: 'LINEITEM' and 'PART'. The 'LINEITEM' tab is active, showing a table with the following columns: ORDERNUMBER, LINEITEMNUMBER, QUANTITY, and PARTNUMBER. The first row contains the values 2, 1, 25, and 456. There are three empty rows below it. At the bottom of the window, it says '17 records found.' and 'Edit Mode'.

ORDERNUMBER	LINEITEMNUMBER	QUANTITY	PARTNUMBER
2	1	25	456

Index

A

- Adaptive Command Interface property 179
 - using 197
 - using for flexible windows 179
- Adaptive Command Set property 177
- Adding supplier plans
 - Application Model Workshop 160
 - Business Model Workshop 98
- Aggregate property 122
- Aggregation 121
 - definition 37, 50, 121
 - deleting records in master-detail classes 121
- Align Classes tools and commands 106
- Align Windows tools and commands 169
- Application
 - compatibility level 231
 - default partitioning for 228
 - deploying 231
 - distributed testing 230
 - supplier project for 224
 - testing 214
- Application model
 - adding supplier plans 160
 - branching 160
 - building 172–214
 - checking out 90, 160
 - components of 156
 - creating 158
 - default behavior for generated windows 208
 - definition 33
 - including callouts 160
 - including multiple business models 160
 - including other application models 160, 200
 - stored in repository 40
 - using callouts 202–207
 - using multiple business models 173
 - validating 209
- Application Model Properties dialog 159, 185
- Application Model Name 159
- Start Window 184
- Application Model Workshop
 - about 156
 - access to other workshops 171
 - accessing 158
 - alignment tools 169
 - Arrange menu 222
 - Component menu 221
 - Edit menu 221
 - editing tools 165
 - File menu 220
 - floatover help description 162
 - leaving 171
 - palette 163
 - preferences 215
 - status line description 162
 - toolbar 164
 - using the window 162
 - View menu 222
 - viewing tools 167

- Application Model Workshop Preferences dialog
 - Attribute List Width 217
 - Diagram Width 216
 - Fonts 219
 - Height 217
 - Initial Wrap 218
 - Position 217
 - Repeat New 218
 - Revert to This Workshop 216
 - X,Y 217
- AppModel project 27
- Architecture 34
- Arrange menu
 - Application Model Workshop 222
 - Business Model Workshop 153
- Array windows
 - available links 187
 - description 175
 - inserting records 240
 - nested 195
 - using 239
- Association Name property 119
- Association Properties dialog
 - Aggregate 122
 - Association Name 119
 - Direction 118
 - From Attribute 119
 - Multiplicity 119
 - To Attribute 119
- Association property 190
- Associations
 - about 117
 - aggregate 50, 121
 - creating 49, 117, 123
 - default direction and multiplicity 122
 - definition 36
 - direction 119, 191
 - From and To attributes 119
 - generalization 125
 - many-to-one 119
 - multiplicity 121
 - one-to-many 120
 - one-to-one 119
 - optional 121

- property sheet 51, 124
- relationship to links 189
- setting properties 124
- Asynchronous windows 193
- Attribute List
 - definition 162
 - using 180
- Attribute List Width property 215
- Attribute Properties dialog 115
 - Column Heading 114
 - Custom 115
 - Database Column 112
 - Domain 113
 - Field Title 114
 - Field Width 113
 - Key 112
- Attribute View Properties dialog 182
- Attributes
 - creating 112
 - custom 115
 - definition 36
 - displaying 62
 - domains 113
 - editing 117
 - in business classes 112
 - modifying order 64
 - not based on database column 115

B

- BooleanDomain 113
- Branch command 40, 91
- Branching
 - a business model 97
 - an application model 160
- Business class properties
 - Class Name 111
 - Database Table 111
 - Send Only Changed Fields on Update 111
- Business class window commands 177

- Business Class Window Properties dialog 180
 - Adaptive Command Interface 179
 - Command Set 177
 - Default Interface 176
 - Direction 175
 - Layout of Fields 174
 - window simulator 180
 - Wrap 176
- Business class windows
 - and the Adaptive Command Interface
 - property 179
 - array window 175
 - commands 177
 - creating 172
 - definition 156
 - displaying fields from two tables 183
 - form window 174
 - layout styles 174
 - outline window 175
 - type of interface 176
- Business Class/Service List
 - definition 100
 - using 110
- Business classes
 - assigning to a service 133
 - associations 117
 - creating 109
 - creating multiple 110
 - definition 36
 - generalization for 125
 - generating customizable 81, 144
 - property sheet 111
 - relationship to database schema 137
 - setting properties 111
- Business Model Name property 97
- Business Model Properties dialog 97
- Business Model Workshop
 - about 95
 - access to other workshops 109
 - accessing 96
 - alignment tools 106
 - Arrange menu 153
 - associations 117
 - Component menu 152
 - Custom property 115
 - Edit menu 151
 - editing tools 102
 - exiting 109
 - File menu 150
 - floatover help description 99
 - New Association Tool 100
 - New Business Class Tool 100
 - New Generalization Tool 100
 - palette 100
 - preferences 145
 - setting preferences for 145
 - status line description 99
 - toolbar 101
 - using 96
 - View menu 153
 - viewing tools 104
 - window 99
- Business Model Workshop Preferences dialog
 - Class List Width 147
 - Diagram Width 147
 - Fonts 149
 - Height 147
 - Max Class Rows 149
 - Position 147
 - Repeat New 149
 - Reset to This Workshop 147
 - X,Y 147
- Business models
 - adding supplier plans 98
 - branching 91, 97
 - checking out 90, 97
 - default service 46, 125
 - definition 33
 - importing database schema 135
 - stored in repository 40
 - validating 140
- Business services, sharing between applications 234
- Button command interface 176
- Button commands 178
- Buttons 176

C

- Called windows
 - definition 186
 - properties 193
- Calling windows 186
- Callout Tool 163, 202
- Callout windows, creating 156202–207
- Callouts
 - links to 206
 - to deployed applications 205
 - to iPlanet UDS projects 203
 - to user-defined methods 202
- Cancel command 109, 171
- Checking out
 - a business model 97
 - an application model 160
- Checkout command 40, 90
- Class List Width property 146
- Class Name property 111
- Class Properties dialog 111
- Client code generation, *See* Code generation, client code
- Client partitions 228
- Close command
 - Application Model Workshop 171
 - Business Model Workshop 109
- Code generation
 - about 209
 - client code 209
 - custom 211
 - definition 38
 - server code 140, 143
- Column Heading property 114
- Command links
 - creating 186
 - using 246
- Command Set property
 - description 177
 - using 69
- Compile All Plans command 93
- Compile All Plans Generates Models preference 93

- Compile Code While Generating property 146, 149
- Component menu
 - Application Model Workshop 221
 - Business Model Workshop 152
- Composite primary keys
 - definition 112
 - reordering 112
- Concurrency properties
 - Database: Explicit Locking 128
 - Database: Native Locking 127
 - None 128
 - Optimistic: Verify 127
- Confirm When Destructive property 208
- Copy tool/command 102, 165
- Custom attributes
 - creating 115
 - definition 115
- Custom Generation Options command
 - Application Model Workshop 211
 - Business Model Workshop 142
 - deleting customizable business classes 144
 - deleting customizable window classes 214
 - effect on future classes in the model 144, 214
- Custom Generation Options dialog
 - Superclass Prefix for Global Customization 142, 212
- Custom property 115
- Customizable classes, naming conventions 86, 144, 213
- Customizing
 - changing the mode of a window 86
 - client code generation 211
 - server code generation 142
 - using templates 81
- Cut tool/command 102, 165

D

- Database Column property 112
- Database Manager property
 - about 129
 - FORTE_EX_DBRESOURCE 132
 - specifying 46

- Database Name property
 - about 129
 - FORTE_EX_DATABASE 132
 - specifying 46
 - Database schema
 - exporting 136
 - importing 135
 - importing from multiple databases 136
 - Database sessions, creating multiple 226
 - Database table definitions, importing into business model 135
 - Database Table property 111
 - Database tables, importing 46
 - Database: Explicit Locking 128
 - Database: Native Locking 127
 - DateTime data, in queries 243
 - DateTimeDomain 113
 - DBService service object 227
 - Default behavior for generated windows
 - Confirm When Destructive 208
 - Maximum Records in Result 208
 - setting 208
 - Default Interface property
 - definition 176
 - using 70
 - Default service 46
 - Delete tool/command 102, 165
 - Deleting records 241
 - Deploying an application 231
 - Detail data, fetching 190
 - Diagram Width property 146, 215
 - Dialog Duration property 128
 - Direction
 - affect on generated classes 49
 - default 122
 - Direction property 118, 119, 175
 - Display property 188
 - Domain property 113
 - Domains
 - custom 116
 - default widgets 113
 - description 113
 - Domain property 113
 - widgets affecting field widths 114
 - DoubleDomain 113
 - Drilldown links
 - definition 186
 - display options 188
 - using 179, 246
- ## E
- Edit menu
 - Application Model Workshop 221
 - Business Model Workshop 151
 - Environment variables 132
 - Equi-join 119
 - Export Database Schema command 139
 - Exporting database schema
 - from multiple services 139
 - relationship between classes and schema 137
 - sample 138
 - express command 28
 - ExpressArrayWindow 211
 - ExpressDBBrowser project 27
 - ExpressDomains project 27
 - ExpressFormWindow 211
 - ExpressOutlineWindow 211
 - ExpressServices project
 - customizing 142
 - required in workspace 27
 - ExpressWindows project
 - customizing 211
 - required in workspace 27
- ## F
- Fetch Dependent Records property 190
 - Fetching detail data 190
 - Field Title property 114
 - Field Width property
 - and domain properties 114
 - description 113
 - Field width, generated fields 114

Fields

- applying a template to 84
- changing width 184
- direction 175
- displaying 62
- label text 114
- modifying order 64
- number to wrap 176
- read only 181
- read/write 181
- resizing 113
- sorting 181

File menu

- Application Model Workshop 220
- Business Model Workshop 150

Find tool/command 103, 165

Flexible windows

- using the Adaptive Command Interface property 179

Folder links

- definition 187
- display options 189
- using 248

Font preferences

- Application Model Workshop 219
- Business Model Workshop 149

Force Compile command 93

Force Complete Generation command

- Application Model Workshop 212
- Business Model Workshop 145

Form windows

- available links 187
- description 174
- inserting records 240
- using 239

forte command 28

FORTE_EX_DBNAME 132

FORTE_EXPRESS_DBPWD 132

FORTE_EXPRESS_DBRESOURCE 132

FORTE_EXPRESS_DBUSER 132

From Attribute property 119

From-class, definition 118

G

Generalizations

- and exported schema 137
- creating 125
- definition 37

Generate Client Code tool/command 164, 214

Generate Server Code command 144

Generate Server Code tool/command 101

Generated Preferences command 208

Generated Preferences dialog

- Confirm When Destructive 208
- Maximum Records in Result 208

Generated windows

- default behavior 208
- modes 237
- using 237

Generating server classes

- forcing complete generation 145
- procedures 52

Generating window classes

- forcing complete generation 212
- procedures 55

Global customization

- ExpressServices project 142
- ExpressWindows project 211

Grid... command 105, 167

H

Height property 146, 215

I

ImageDomain 113

Import Database Schema command

- importing from multiple databases 136
- support for primary keys 48
- using 46, 135

Included application windows 156
 Inheritance 125
 Initial Adaptive Command Interface property 216
 Initial Wrap property 216, 218
 Inserting into nested windows 195
 Inserting records in windows 240
 IntegerDomain 113

K

Key property 112

L

Label property 187
 Layout of Fields property 174
 Link Properties dialog

- Association 190
- Display 188
- Label 187
- mode status 193
- Pre-Fetch Dependent Records 190
- read status 193
- Type 187
- using 194
- window simulator 194

 Links

- Association property 190
- between classes from different models 190
- command 186, 246
- creating 186
- definition 157
- Display property 188
- drilldown 186, 246
- Fetch Dependent Records property 190
- folder (tab folder) 187, 248
- Label property 187
- lookup 187, 247
- multiple 197
- nested 186, 248
- relationship to associations 189

- to a callout 206
- to included application model 202
- Type property 187
- types of 186
- using 67, 245

LongTextDomain 113
 Lookup links

- creating 69
- definition 187
- display options 189
- using 247

M

Main project 224
 Max Class Rows property 146, 149
 Maximum Records in Result property 208
 Menu and Toolbar command interface 176
 Menu commands 177
 Mode property 193
 MoneyDomain 113
 Multiple database sessions

- creating 129, 226

 Multiplicity

- about 119
- affect on generated classes 49
- default 122
- definition 37, 119

 Multiplicity property 121

N

Nested links

- about 194
- creating 73
- definition 186
- display options 189
- using 248

 Nested windows 194–196

- New Application Model tool/command 102, 159
- New Association tool
 - about 100
 - using 49, 123
- New Attribute tool/command 101, 115
- New Callout Tool 202
- New Class tool/command 110
- New Class Window tool/command 163, 173
- New Included Application Model tool 163, 200
- New Link Tool 164
- New Reference Partition command 235

O

- Object Modeling Technique 35
- Optimistic: Verify 127
- Optional association 119, 121
- Outline windows
 - available links 187
 - description 175
 - editing 178
 - inserting into 179
 - inserting records 240
 - nested 194
 - using 239

P

- Partition Workshop
 - deploying an application 231
 - modifying a configuration 232
 - testing an application 230
- Partitioning, default 39, 228
- Partitions
 - client 228
 - replicating 234
 - router partition 234
 - server 228
- Paste tool/command 102, 165
- PDF files, viewing and searching 24

- Plans, compiling 93
- Position property 146, 215
- Preferences
 - Application Model Workshop 215
 - Business Model Workshop 145
- Primary keys
 - and exported schema 137
 - composite 112
 - handled by Import Database Schema command 48
 - setting Key property 112
- Print tool/command 102, 165
- Projects, created by Express 224
- Properties command
 - Application Model Workshop 180
 - Business Model Workshop 111
 - description 164

Q

- Query by example
 - searching date time fields 242
 - searching numeric data fields 242
 - searching text fields 241
 - using 57

R

- Read status property 181
- Read/Write property 193
- Read-Only property 193
- Records
 - deleting 241
 - inserting 240
 - scrolling 244
 - selecting 241
- Redo tool/command 103, 165
- Reference partitions 234
- Regenerating server classes 145

- Regeneration
 - forcing complete (Application Model) 212
 - forcing complete (Business Model) 145
- Repeat New command 110, 124, 173
- Repeat New property 146, 215
- Replication partitions 234
- Repository Workshop
 - Compile All Plans command 93
 - Force Compile command 93
- Reset to This Workshop property 146, 216
- Result sets
 - clearing 244
 - definition 237
 - saving changes to the database 245
- Revert to Last Saved command 245
- Router partitions 234
- Run Application tool/command 164, 214

S

- Save All tool/command 102, 165
- Search criteria 242
- Search mode 238
- Select Tool
 - Application Model Workshop 163
 - Business Model Workshop 100
- Selecting records
 - Query by example 241
 - using wildcards 241
- Send Only Changed Fields on Update property 111
- Server classes
 - forcing complete generation 145
 - generating 52
- Server code generation, *See* Code generation, server code
- Server partitions 228
- Service Name property 127
- Service objects
 - about 224
 - DBService 227
 - modifying definitions 232
 - moving 233
 - Service 226
 - TheBroker 225
 - unassigned 229
- Service properties
 - setting 46
- Service Properties dialog
 - Concurrency 127
 - Database Manager 129
 - Database Name 129
 - Dialog Duration 128
 - Password 129
 - Service Name 127
 - User Name 129
 - Visibility 128
- Service service object 226
- Services
 - assigning a class to a 133
 - creating 129
 - creating multiple database sessions 129
 - default 125
 - definition 46
 - deleting 134
 - environment variables 132
- Sort direction property 181
- Sort order property 181
- Start Window property 184
- Start windows
 - default 158, 184
 - definition 158
 - specifying 184
- starting Express 28
- Superclass Prefix for Global Customization
 - Application Model Workshop 212
 - Business Model Workshop 142
- Supplier plans
 - and generated projects 98, 161
 - Application Model Workshop 160
 - Business Model Workshop 98
 - command 98, 161
 - to included application models 202
- Supplier projects 224
- Synchronous windows 193

T

- Tab folders
 - See also* Folder links
- Templates, for data fields 81
- Testing
 - from Application Model Workshop 214
 - from Partition Workshop 230
 - Partition Workshop 40
- Text Tool
 - Application Model Workshop 163
 - Business Model Workshop 100
- TextDomain 113
- TheBroker service object 225
- Three-tier architecture 34
- To Attribute property 119
- To-class, definition 118
- Toolbar commands 178
- Tutorial, sample table descriptions 44
- Type property 187

U

- Undo Checkout/Branch command 92
- Undo Deleted Checkouts command 92
- Undo tool/command 103, 165
- User Name property
 - FORTE_EX_DBUSER 132
 - specifying 46
- User Password property
 - FORTE_EX_DBPWD 132
 - specifying 46

V

- Validate Model command
 - Application Model Workshop 209
 - Business Model Workshop 140
 - error messages (application model) 209
 - error messages (business model) 141
- View menu
 - Application Model Workshop 222
 - Business Model Workshop 153
 - setting default preferences 148, 218
 - using 107, 170
- View property 146, 215
- Visibility property 128

W

- Wildcard character 57, 241
- Window attributes
 - properties of 180
 - read status 181
 - sort direction 181
 - sort order 181
- Window classes
 - forcing complete generation 212
 - generating 55
 - generating customizable 81, 213
- Window fields
 - read status 181
 - sort direction 181
 - sort order 181
 - specifying 180
- Window layout styles 174
- Window simulator
 - Business Class Window Properties dialog 180
 - Link properties dialog 194

Windows

- absolute window position 147, 217
- array style 175
- asynchronous 193
- attributes, *See* Window attributes
- behavior of 157
- classes, *See* Window classes
- commands 177
- displaying fields from two tables 183
- fields, *See* Window fields
- form style 174
- layout styles, *See* Window layout styles
- links for 157
- modal 193
- modeless 193
- modes 86, 237
- outline style 175
- start window 158
- styles of 156
- using 237
- using the Adaptive Command Interface
 - property 179
- X,Y property 146, 215

- Workshop Preferences command
 - Application Model Workshop 215
 - Business Model Workshop 145
- workshops
 - before using 27
 - starting 28
- Wrap property 176

X

- X,Y property 146, 215

Z

- Zoom tools/commands 104, 167

