OpenEdge® Getting Started:
Progress OpenEdge Studio
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preface</td>
<td>Preface–1</td>
</tr>
<tr>
<td>1. Overview of the OpenEdge Studio</td>
<td>1–1</td>
</tr>
<tr>
<td>Starting the OpenEdge Desktop</td>
<td>1–2</td>
</tr>
<tr>
<td>Starting a tool from the Desktop</td>
<td>1–2</td>
</tr>
<tr>
<td>Starting a tool from another tool</td>
<td>1–3</td>
</tr>
<tr>
<td>OpenEdge Studio tools</td>
<td>1–4</td>
</tr>
<tr>
<td>OpenEdge online help system</td>
<td>1–6</td>
</tr>
<tr>
<td>Overview of the OpenEdge AppBuilder</td>
<td>1–7</td>
</tr>
<tr>
<td>AppBuilder main window</td>
<td>1–7</td>
</tr>
<tr>
<td>AppBuilder Object Palette</td>
<td>1–11</td>
</tr>
<tr>
<td>Advanced editing features in OpenEdge Studio</td>
<td>1–15</td>
</tr>
<tr>
<td>2. Working with the Sample Application</td>
<td>2–1</td>
</tr>
<tr>
<td>The SportsPro sample application</td>
<td>2–2</td>
</tr>
<tr>
<td>Creating and connecting to the Sports2000 database</td>
<td>2–3</td>
</tr>
<tr>
<td>Copying the sample files</td>
<td>2–6</td>
</tr>
<tr>
<td>Compiling the sample files</td>
<td>2–7</td>
</tr>
<tr>
<td>Running the sample application</td>
<td>2–8</td>
</tr>
<tr>
<td>Dynamic Query module</td>
<td>2–10</td>
</tr>
<tr>
<td>Dynamic Buffer/Temp Table module</td>
<td>2–12</td>
</tr>
<tr>
<td>Drag and Drop module</td>
<td>2–14</td>
</tr>
<tr>
<td>Resizable Dynamic Browser</td>
<td>2–15</td>
</tr>
<tr>
<td>SmartDataObject module</td>
<td>2–17</td>
</tr>
<tr>
<td>SmartDataField module</td>
<td>2–19</td>
</tr>
<tr>
<td>Publish and Subscribe module</td>
<td>2–20</td>
</tr>
<tr>
<td>SmartB2B and SonicMQ module</td>
<td>2–22</td>
</tr>
<tr>
<td>Working with the Order Tracking System</td>
<td>2–26</td>
</tr>
<tr>
<td>Business scenario</td>
<td>2–26</td>
</tr>
<tr>
<td>Customer section</td>
<td>2–29</td>
</tr>
<tr>
<td>Order section</td>
<td>2–33</td>
</tr>
<tr>
<td>Order Line section</td>
<td>2–39</td>
</tr>
<tr>
<td>Menu items</td>
<td>2–40</td>
</tr>
</tbody>
</table>
Figures

Figure 1–1: OpenEdge desktop ................................................................. 1–2
Figure 1–2: AppBuilder main window and Object Palette—initial display .... 1–7
Figure 1–3: The AppBuilder main window .............................................. 1–8
Figure 1–4: The AppBuilder Object Palette ........................................... 1–11
Figure 4–1: SmartObjects in the Order Tracking System ......................... 4–3
Figure 4–2: SmartObject types and SmartLinks ...................................... 4–5
Figure 4–3: SmartFolder and Page SmartLink ......................................... 4–6
Figure 4–4: SmartLinks for dcust SDO ................................................ 4–6
Figure 4–5: SmartLinks for dorder SDO ................................................ 4–7
Figure 4–6: SmartLinks for dordline SDO ............................................... 4–7
<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1–1</td>
<td>Toolbar icons</td>
<td>1–9</td>
</tr>
<tr>
<td>1–2</td>
<td>Object Palette icons</td>
<td>1–12</td>
</tr>
<tr>
<td>2–1</td>
<td>Data Dictionary icons</td>
<td>2–5</td>
</tr>
<tr>
<td>2–2</td>
<td>Components of the SportsPro main window</td>
<td>2–9</td>
</tr>
<tr>
<td>2–3</td>
<td>Components of the Dynamic Query module</td>
<td>2–11</td>
</tr>
<tr>
<td>2–4</td>
<td>Components of the Dynamic Buffer/Temp Table module</td>
<td>2–13</td>
</tr>
<tr>
<td>2–5</td>
<td>Components of the Drag and Drop module</td>
<td>2–15</td>
</tr>
<tr>
<td>2–6</td>
<td>Components of the Resizable Dynamic Browser module</td>
<td>2–16</td>
</tr>
<tr>
<td>2–7</td>
<td>Components of the SmartDataObject module</td>
<td>2–18</td>
</tr>
<tr>
<td>2–8</td>
<td>Components of the SmartDataField module</td>
<td>2–20</td>
</tr>
<tr>
<td>2–9</td>
<td>Components of the Publish and Subscribe module</td>
<td>2–21</td>
</tr>
<tr>
<td>2–10</td>
<td>Components of the SmartB2B and SonicMQ module</td>
<td>2–24</td>
</tr>
<tr>
<td>2–11</td>
<td>Components of the Order Tracking System main window</td>
<td>2–27</td>
</tr>
<tr>
<td>2–12</td>
<td>Components of the Maintain Customer section</td>
<td>2–32</td>
</tr>
<tr>
<td>2–13</td>
<td>Components of the Order Entry / Maintenance section</td>
<td>2–35</td>
</tr>
<tr>
<td>2–14</td>
<td>Components of the Find Order section</td>
<td>2–38</td>
</tr>
<tr>
<td>2–15</td>
<td>Components of the Inventory Transfer section</td>
<td>2–41</td>
</tr>
<tr>
<td>2–16</td>
<td>Components of the Set Application Language section</td>
<td>2–42</td>
</tr>
<tr>
<td>4–1</td>
<td>AppBuilder filename conventions</td>
<td>4–8</td>
</tr>
<tr>
<td>4–2</td>
<td>Information in the documentation set</td>
<td>4–9</td>
</tr>
</tbody>
</table>
Preface

This Preface contains the following sections:

• Purpose
• Audience
• Organization
• Using this manual
• Typographical conventions
• OpenEdge messages
• Third party acknowledgements
Purpose

This guide introduces you to the OpenEdge Studio that you can use to easily develop and maintain applications. It briefly introduces the Application Development Environment (ADE) Desktop and toolset, leads you through a sample application that shows some of what you can do with the tools, and presents a tutorial that shows you how to use the OpenEdge AppBuilder to create part of the sample application.

Audience

This guide is for developers using the OpenEdge Studio ADE. This guide contains valuable information for developers familiar with ABL as well as developers who are new to ABL and the ADE.

Organization

This guide provides an introduction to the OpenEdge Studio and SmartObject technology.

Chapter 1, “Overview of the OpenEdge Studio”

Provides an overview of the OpenEdge Desktop and introduces you to the tools that comprise the ADE. In addition to a general description of the ADE, this chapter provides a more detailed description of the AppBuilder.

Chapter 2, “Working with the Sample Application”

Provides information about the SportsPro sample application. By working with the sample application, you can better understand the ABL and the AppBuilder. This chapter also details the structure of each module in the sample application.

Chapter 3, “Assembling the Order Tracking System”

Provides exercises for assembling the main window of the SportsPro Order Tracking System from a set of SmartObjects supplied with the OpenEdge Studio. Assembling the main window provides an understanding of how to work with SmartObjects.

Chapter 4, “Overview of SmartObject Technology”

Provides an introduction to SmartObjects and how they connect to each other to form applications.

Using this manual

This document assumes that you have successfully installed the OpenEdge Studio. The information in it is designed to acquaint you with the fundamental tools and techniques for working with ABL in the OpenEdge Studio. The exercises in Chapter 3, “Assembling the Order Tracking System,” are arranged in three sections: an overview of the techniques needed to complete the exercise, an abbreviated procedures for those who want to try the exercise with minimal direction, and a more detailed procedure for more help.
OpenEdge provides a special purpose programming language for building business applications. In the documentation, the formal name for this language is ABL (Advanced Business Language). With few exceptions, all keywords of the language appear in all UPPERCASE, using a font that is appropriate to the context. All other alphabetic language content appears in mixed case.

For the latest documentation updates see the OpenEdge Product Documentation category on PSDN http://www.psdn.com/library/kbcategory.jspa?categoryID=129.

References to ABL compiler and run-time features

ABL is both a compiled and interpreted language that executes in a run-time engine that the documentation refers to as the ABL Virtual Machine (AVM). When documentation refers to ABL source code compilation, it specifies ABL or the compiler as the actor that manages compile-time features of the language. When documentation refers to run-time behavior in an executing ABL program, it specifies the AVM as the actor that manages the specified run-time behavior in the program.

For example, these sentences refer to the ABL compiler’s allowance for parameter passing and the AVM’s possible response to that parameter passing at run time: “ABL allows you to pass a dynamic temp-table handle as a static temp-table parameter of a method. However, if at run time the passed dynamic temp-table schema does not match the schema of the static temp-table parameter, the AVM raises an error.” The following sentence refers to run-time actions that the AVM can perform using a particular ABL feature: “The ABL socket object handle allows the AVM to connect with other ABL and non-ABL sessions using TCP/IP sockets.”

References to ABL data types

ABL provides built-in data types, pre-defined class data types, and user-defined class data types. References to built-in data types follow these rules:

• Like most other keywords, references to specific built-in data types appear in all UPPERCASE, using a font that is appropriate to the context. No uppercase reference ever includes or implies any data type other than itself.

• Wherever integer appears, this is a reference to the INTEGER or INT64 data type.

• Wherever decimal appears, this is a reference to the DECIMAL data type.

• Wherever numeric appears, this is a reference to the INTEGER, INT64, or DECIMAL data type.

References to pre-defined class data types appear in mixed case with initial caps, for example, Progress.Lang.Object. References to user-defined class data types appear in mixed case, as specified for a given application example.
Typographical conventions

This manual uses the following typographical conventions:

<table>
<thead>
<tr>
<th>Convention</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bold</td>
<td>Bold typeface indicates commands or characters the user types, or the names of user interface elements.</td>
</tr>
<tr>
<td>Italic</td>
<td>Italic typeface indicates the title of a document, provides emphasis, or signifies new terms.</td>
</tr>
<tr>
<td>SMALL, BOLD CAPITAL LETTERS</td>
<td>Small, bold capital letters indicate OpenEdge™ key functions and generic keyboard keys; for example, GET and CTRL.</td>
</tr>
<tr>
<td>KEY1-KEY2</td>
<td>A hyphen between key names indicates a simultaneous key sequence: you press and hold down the first key while pressing the second key. For example, CTRL-X.</td>
</tr>
<tr>
<td>KEY1 KEY2</td>
<td>A space between key names indicates a sequential key sequence: you press and release the first key, then press another key. For example, ESCAPE H.</td>
</tr>
</tbody>
</table>

Syntax:

| Fixed width                     | A fixed-width font is used in syntax statements, code examples, and for system output and filenames. |
| Fixed-width italics             | Fixed-width italics indicate variables in syntax statements. |
| Fixed-width bold                | Fixed-width bold indicates variables with special emphasis. |
| UPPERCASE fixed width           | Uppercase words are ABL language keywords. Although these always are shown in uppercase, you can type them in either uppercase or lowercase in a procedure. |

Example procedures

This manual provides numerous example procedures that illustrate syntax and concepts. You can access the example files and details for installing the examples from the following locations:

- The Documentation and Samples CD that you received with your product.
- The OpenEdge Documentation page on PSDN:

OpenEdge messages

OpenEdge displays several types of messages to inform you of routine and unusual occurrences:

- **Execution messages** inform you of errors encountered while OpenEdge is running a procedure; for example, if OpenEdge cannot find a record with a specified index field value.

- **Compile messages** inform you of errors found while OpenEdge is reading and analyzing a procedure before running it; for example, if a procedure references a table name that is not defined in the database.

- **Startup messages** inform you of unusual conditions detected while OpenEdge is getting ready to execute; for example, if you entered an invalid startup parameter.

After displaying a message, OpenEdge proceeds in one of several ways:

- Continues execution, subject to the error-processing actions that you specify or that are assumed as part of the procedure. This is the most common action taken after execution messages.

- Returns to the Progress Procedure Editor, so you can correct an error in a procedure. This is the usual action taken after compiler messages.

- Halts processing of a procedure and returns immediately to the Procedure Editor. This does not happen often.

- Terminates the current session.

OpenEdge messages end with a message number in parentheses. In this example, the message number is 200:

```
** Unknown table name table. (200)
```

If you encounter an error that terminates OpenEdge, note the message number before restarting.

Obtaining more information about OpenEdge messages

On Windows platforms, use OpenEdge online help to obtain more information about OpenEdge messages. Many OpenEdge tools include the following Help menu options to provide information about messages:

- Choose **Help → Recent Messages** to display detailed descriptions of the most recent OpenEdge message and all other messages returned in the current session.

- Choose **Help → Messages** and then enter the message number to display a description of a specific OpenEdge message.

- In the Procedure Editor, press the HELP key or F1.
Third party acknowledgements

OpenEdge includes AdventNet - Agent Toolkit licensed from AdventNet, Inc. http://www.adventnet.com. All rights to such copyright material rest with AdventNet.

OpenEdge includes ANTLR (Another Tool for Language Recognition) software Copyright © 2003-2006, Terence Parr All rights reserved. Neither the name of the author nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission. Software distributed on an “AS IS” basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License agreement that accompanies the product.

OpenEdge includes software developed by the Apache Software Foundation (http://www.apache.org/). Copyright © 1999 The Apache Software Foundation. All rights reserved (Xerces C++ Parser (XML) and Xerces2 Java Parser (XML)); Copyright © 1999-2002 The Apache Software Foundation. All rights reserved (Xerces Parser (XML)); and Copyright © 2000-2003 The Apache Software Foundation. All rights reserved (Ant). The names “Apache,” “Xerces,” “ANT,” and “Apache Software Foundation” must not be used to endorse or promote products derived from this software without prior written permission. Products derived from this software may not be called “Apache”, nor may “Apache” appear in their name, without prior written permission of the Apache Software Foundation. For written permission, please contact apache@apache.org. Software distributed on an “AS IS” basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License agreement that accompanies the product.

OpenEdge includes Concurrent Java software Copyright 1994-2000 Sun Microsystems, Inc. All Rights Reserved. -Neither the name of or trademarks of Sun may be used to endorse or promote products including or derived from the Java Software technology without specific prior written permission; and Redistributions of source or binary code must contain the above copyright notice, this notice and the following disclaimers: This software is provided "AS IS," without a warranty of any kind. ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE HEREBY EXCLUDED. SUN MICROSYSTEMS, INC. AND ITS LICENSORS SHALL NOT BE LIABLE FOR ANY DAMAGES SUFFERED BY LICENSEE AS A RESULT OF USING, MODIFYING OR DISTRIBUTING THE SOFTWARE OR ITS DERIVATIVES. IN NO EVENT WILL SUN MICROSYSTEMS, INC. OR ITS LICENSORS BE LIABLE FOR ANY LOST REVENUE, PROFIT OR DATA, OR FOR DIRECT, INDIRECT, SPECIAL, CONSEQUENTIAL, INCIDENTAL OR PUNITIVE DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF THE USE OF OR INABILITY TO USE SOFTWARE, EVEN IF SUN MICROSYSTEMS, INC. HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

OpenEdge includes DataDirect software Copyright © 1991-2007 Progress Software Corporation and/or its subsidiaries or affiliates. All Rights Reserved. (DataDirect Connect for JDBC Type 4 driver); Copyright © 1993-2009 Progress Software Corporation and/or its subsidiaries or affiliates. All Rights Reserved. (DataDirect Connect for JDBC); Copyright © 1988-2007 Progress Software Corporation and/or its subsidiaries or affiliates. All Rights Reserved. (DataDirect Connect for ODBC); and Copyright © 1988-2007 Progress Software
Corporation and/or its subsidiaries or affiliates. All Rights Reserved. (DataDirect Connect64 for ODBC).

OpenEdge includes DataDirect Connect for ODBC and DataDirect Connect64 for ODBC software, which include ICU software 1.8 and later - Copyright © 1995-2003 International Business Machines Corporation and others All rights reserved. Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, provided that the above copyright notice(s) and this permission notice appear in all copies of the Software and that both the above copyright notice(s) and this permission notice appear in supporting documentation.

OpenEdge includes DataDirect Connect for ODBC and DataDirect Connect64 for ODBC software, which include software developed by the OpenSSL Project for use in the OpenSSL Toolkit (http://www.openssl.org/). Copyright © 1998-2006 The OpenSSL Project. All rights reserved. And Copyright © 1995-1998 Eric Young (eay@cryptsoft.com). All rights reserved.

OpenEdge includes DataDirect products for the Microsoft SQL Server database which contain a licensed implementation of the Microsoft TDS Protocol.

OpenEdge includes software authored by David M. Gay. Copyright © 1991, 2000, 2001 by Lucent Technologies (dtoa.c); Copyright © 1991, 1996 by Lucent Technologies (g_fmt.c); and Copyright © 1991 by Lucent Technologies (rnd_prod.s). Permission to use, copy, modify, and distribute this software for any purpose without fee is hereby granted, provided that this entire notice is included in all copies of any software which is or includes a copy or modification of this software and in all copies of the supporting documentation for such software. THIS SOFTWARE IS BEING PROVIDED "AS IS", WITHOUT ANY EXPRESS OR IMPLIED WARRANTY. IN PARTICULAR, NEITHER THE AUTHOR NOR LUCENT MAKES ANY REPRESENTATION OR WARRANTY OF ANY KIND CONCERNING THE MERCHANTABILITY OF THIS SOFTWARE OR ITS FITNESS FOR ANY PARTICULAR PURPOSE.

OpenEdge includes software authored by David M. Gay. Copyright © 1998-2001 by Lucent Technologies All Rights Reserved (decstrtod.c; strtodg.c); Copyright © 1998, 2000 by Lucent Technologies All Rights Reserved (decstrtof.c; strtord.c); Copyright © 1998 by Lucent Technologies All Rights Reserved (dmisc.c; gdtoa.h; gethex.c; gmisc.c; sum.c); Copyright © 1998, 1999 by Lucent Technologies All Rights Reserved (gdtoa.c; misc.c; smisc.c; ulp.c); Copyright © 1998-2000 by Lucent Technologies All Rights Reserved (gdtoaimp.h); Copyright © 2000 by Lucent Technologies All Rights Reserved (hd_init.c). Full copies of these licenses can be found in the installation directory, in the c:/OpenEdge/licenses folder. Permission to use, copy, modify, and distribute this software and its documentation for any purpose and without fee is hereby granted, provided that the above copyright notice appear in all copies and that both that the copyright notice and this permission notice and warranty disclaimer appear in supporting documentation, and that the name of Lucent or any of its entities not be used in advertising or publicity pertaining to distribution of the software without specific, written prior permission. LUCENT DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE, INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL LUCENT OR ANY OF ITS ENTITIES BE LIABLE FOR ANY SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
OpenEdge includes http package software developed by the World Wide Web Consortium. Copyright © 1994-2002 World Wide Web Consortium, (Massachusetts Institute of Technology, European Research Consortium for Informatics and Mathematics, Keio University). All rights reserved. This work is distributed under the W3C® Software License [http://www.w3.org/Consortium/Legal/2002/copyright-software-20021231] in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

OpenEdge includes ICU software 1.8 and later - Copyright © 1995-2003 International Business Machines Corporation and others All rights reserved. Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, provided that the above copyright notice(s) and this permission notice appear in all copies of the Software and that both the above copyright notice(s) and this permission notice appear in supporting documentation.


OpenEdge includes Infragistics NetAdvantage for .NET v2009 Vol 2 Copyright © 1996-2009 Infragistics, Inc. All rights reserved.

OpenEdge includes JSTL software Copyright 1994-2006 Sun Microsystems, Inc. All Rights Reserved. Software distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License agreement that accompanies the product.

OpenEdge includes OpenSSL software developed by the OpenSSL Project for use in the OpenSSL Toolkit (http://www.openssl.org/). Copyright © 1998-2007 The OpenSSL Project. All rights reserved. This product includes cryptographic software written by Eric Young (eay@cryptsoft.com). This product includes software written by Tim Hudson (tjh@cryptsoft.com). Copyright © 1995-1998 Eric Young (eay@cryptsoft.com) All rights reserved. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact openssl-core@openssl.org. Products derived from this software may not be called "OpenSSL" nor may "OpenSSL" appear in their names without prior written permission of the OpenSSL Project. Software distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License agreement that accompanies the product.

OpenEdge includes Quartz Enterprise Job Scheduler software Copyright © 2001-2003 James House. All rights reserved. Software distributed on an “AS IS” basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License agreement that accompanies the product. This product uses and includes within its distribution, software developed by the Apache Software Foundation (http://www.apache.org/).

OpenEdge includes code licensed from RSA Security, Inc. Some portions licensed from IBM are available at http://oss.software.ibm.com/icu4j/.

OpenEdge includes the RSA Data Security, Inc. MD5 Message-Digest Algorithm. Copyright ©1991-2, RSA Data Security, Inc. Created 1991. All rights reserved.
OpenEdge includes Sonic software, which includes software developed by Apache Software Foundation (http://www.apache.org/). Copyright © 1999-2000 The Apache Software Foundation. All rights reserved. The names “Ant”, “Axis”, “Xalan”, “FOP”, “The Jakarta Project”, “Tomcat”, “Xerces” and/or “Apache Software Foundation” must not be used to endorse or promote products derived from the Product without prior written permission. Any product derived from the Product may not be called “Apache”, nor may “Apache” appear in their name, without prior written permission. For written permission, please contact apache@apache.org.

OpenEdge includes Sonic software, which includes software developed by ExoLab Project (http://www.exolab.org/). Copyright © 2000 Intalio Inc. All rights reserved. The names “Castor” and/or “ExoLab” must not be used to endorse or promote products derived from the Products without prior written permission. For written permission, please contact info@exolab.org. Exolab, Castor and Intalio are trademarks of Intalio Inc.

OpenEdge includes Sonic software, which includes software developed by IBM. Copyright © 1995-2003 International Business Machines Corporation and others. All rights reserved. Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, provided that the above copyright notice(s) and this permission notice appear in all copies of the Software and that both the above copyright notice(s) and this permission notice appear in supporting documentation. Software distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License agreement that accompanies the product. Except as contained in this notice, the name of a copyright holder shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Software without prior written authorization of the copyright holder.

OpenEdge includes Sonic software, which includes the JMX Technology from Sun Microsystems, Inc. Use and Distribution is subject to the Sun Community Source License available at http://sun.com/software/communitysource.

OpenEdge includes Sonic software, which includes software developed by the ModelObjects Group (http://www.modelobjects.com). Copyright © 2000-2001 ModelObjects Group. All rights reserved. The name “ModelObjects” must not be used to endorse or promote products derived from this software without prior written permission. Products derived from this software may not be called “ModelObjects”, nor may “ModelObjects” appear in their name, without prior written permission. For written permission, please contact djacobs@modelobjects.com.

OpenEdge includes Sonic software, which includes code licensed from Mort Bay Consulting Pty. Ltd. The Jetty Package is Copyright © 1998 Mort Bay Consulting Pty. Ltd. (Australia) and others.
OpenEdge includes Sonic software, which includes files that are subject to the Netscape Public License Version 1.1 (the “License”); you may not use this file except in compliance with the License. You may obtain a copy of the License at http://www.mozilla.org/NPL/. Software distributed under the License is distributed on an “AS IS” basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License. The Original Code is Mozilla Communicator client code, released March 31, 1998. The Initial Developer of the Original Code is Netscape Communications Corporation. Portions created by Netscape are Copyright 1998-1999 Netscape Communications Corporation. All Rights Reserved.

OpenEdge includes Sonic software, which includes software developed by the University Corporation for Advanced Internet Development http://www.ucaid.edu Internet2 Project. Copyright © 2002 University Corporation for Advanced Internet Development, Inc. All rights reserved. Neither the name of OpenSAML nor the names of its contributors, nor Internet2, nor the University Corporation for Advanced Internet Development, Inc., nor UCAID may be used to endorse or promote products derived from this software and products derived from this software may not be called OpenSAML, Internet2, UCAID, or the University Corporation for Advanced Internet Development, nor may OpenSAML appear in their name without prior written permission of the University Corporation for Advanced Internet Development. For written permission, please contact opensaml@opensaml.org.

OpenEdge includes the UnixWare platform of Perl Runtime authored by Kiem-Phong Vo and David Korn. Copyright © 1991, 1996 by AT&T Labs. Permission to use, copy, modify, and distribute this software for any purpose without fee is hereby granted, provided that this entire notice is included in all copies of any software which is or includes a copy or modification of this software and in all copies of the supporting documentation for such software. THIS SOFTWARE IS BEING PROVIDED “AS IS”, WITHOUT ANY EXPRESS OR IMPLIED WARRANTY. IN PARTICULAR, NEITHER THE AUTHORS NOR AT&T LABS MAKE ANY REPRESENTATION OR WARRANTY OF ANY KIND CONCERNING THE MERCHANTABILITY OF THIS SOFTWARE OR ITS FITNESS FOR ANY PARTICULAR PURPOSE.

OpenEdge includes Vermont Views Terminal Handling Package software developed by Vermont Creative Software. Copyright © 1988-1991 by Vermont Creative Software.

OpenEdge includes XML Tools, which includes versions 8.9 of the Saxon XSLT and XQuery Processor from Saxonica Limited (http://www.saxonica.com/) which are available from SourceForge (http://sourceforge.net/projects/saxon/). The Original Code of Saxon comprises all those components which are not explicitly attributed to other parties. The Initial Developer of the Original Code is Michael Kay. Until February 2001 Michael Kay was an employee of International Computers Limited (now part of Fujitsu Limited), and original code developed during that time was released under this license by permission from International Computers Limited. From February 2001 until February 2004 Michael Kay was an employee of Software AG, and code developed during that time was released under this license by permission from Software AG, acting as a "Contributor". Subsequent code has been developed by Saxonica Limited, of which Michael Kay is a Director, again acting as a "Contributor". A small number of modules, or enhancements to modules, have been developed by other individuals (either written especially for Saxon, or incorporated into Saxon having initially been released as part of another open source product). Such contributions are acknowledged individually in comments attached to the relevant code modules. All Rights Reserved. The contents of the Saxon files are subject to the Mozilla Public License Version 1.0 (the "License"); you may not use these files except in compliance with the License. You may obtain a copy of the License at http://www.mozilla.org/MPL/ and a copy of the license can also be found in the
installation directory, in the c:/OpenEdge/licenses folder. Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

OpenEdge includes XML Tools, which includes Xs3P v1.1.3. The contents of this file are subject to the DSTC Public License (DPL) Version 1.1 (the "License"); you may not use this file except in compliance with the License. A copy of the license can be found in the installation directory, in the c:/OpenEdge/licenses folder. Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License. The Original Code is xs3p. The Initial Developer of the Original Code is DSTC. Portions created by DSTC are Copyright © 2001, 2002 DSTC Pty Ltd. All rights reserved.

OpenEdge includes YAJL software Copyright 2007, Lloyd Hilaiel. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met: 1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer. 2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. 3. Neither the name of Lloyd Hilaiel nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission. THIS SOFTWARE IS PROVIDED BY THE AUTHOR ` ` AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
Overview of the OpenEdge Studio

The OpenEdge Studio provides access to a variety of tools that help you develop applications. This chapter provides an overview of the OpenEdge Desktop and introduces you to the tools that comprise the OpenEdge Studio. This chapter also provides a more detailed description of the OpenEdge AppBuilder in the following sections:

- Starting the OpenEdge Desktop
- OpenEdge Studio tools
- OpenEdge online help system
- Overview of the OpenEdge AppBuilder
Starting the OpenEdge Desktop

The OpenEdge Desktop provides access to a number of tools.

To start the Desktop, choose Programs→OpenEdge→Desktop from the Windows Start menu. A Desktop window similar to the one shown in Figure 1–1 opens.

Figure 1–1: OpenEdge desktop

Note: The contents and appearance of the Desktop depend upon the tools licensed and installed at your site. You might see more or fewer tools than those shown in Figure 1–1.

The Desktop is made up of the following:

- **Menu bar** — Provides access to the available Desktop menus. You use the Desktop menus to perform file operations, to start OpenEdge tools, to minimize or maximize the Desktop, to view messages, and to access online help.

  See the “OpenEdge online help system” section on page 1–6 for a description of the selections on the Help menu.

- **Toolbar** — Provides access to a set of icons that allow you to easily access your OpenEdge Studio tools. The individual tools are discussed in the “OpenEdge Studio tools” section on page 1–4.

  Note: If you leave your cursor over an icon, a ToolTip appears naming the tool.

Starting a tool from the Desktop

You can start a tool from the Desktop using either of the following methods:

- Clicking once on its icon

- Choosing a tool from the Tools menu
Starting a tool from another tool

Most of the individual tools have their own Tools menu. When a tool has its own Tools menu, you can run other tools from that tool without returning to the Desktop. For example, the AppBuilder has a Tools menu that includes the Data Dictionary option. As a result, you can run the OpenEdge Data Dictionary from the AppBuilder.

Accessing tools from other tools this way helps you manage the number of tools running (you cannot have more than one instance of a tool running within a session), and enforces the modality between the tools you run. For example, if you run the Data Dictionary from the AppBuilder, you must exit the Data Dictionary before returning to the AppBuilder.

An exception to this behavior is PRO*Tools. When you run PRO*Tools, it displays as a palette of icons that are available for use at any time. See the “OpenEdge Studio tools” section on page 1–4 for additional information.
OpenEdge Studio tools

The tool descriptions are listed in the order in which they appear in the Tools menu. An asterisk (*) indicates that you can also access the tool from a tool icon. If a tool has not been licensed or installed at your site, its icon does not appear in the Desktop.

The OpenEdge Studio tools are:

- **Data dictionary*** — A tool used to define the structure of your database including tables, fields, and indexes, and to generate reports about the database. You can use this tool to define application defaults that, for example, validate data, perform security checks, and generate unique sequence numbers. ABL (Advanced Business Language) gives you the flexibility to override most of these defaults in individual applications.

  You can also use the Data Dictionary to create and connect to databases.

- **Procedure editor*** — An editor used to create, edit, compile, and run unstructured ABL procedures (as opposed to the structured procedures created by the AppBuilder). The Procedure Editor also provides functionality for checking syntax, managing multiple files, and customizing its environment.

- **Data administration** — A tool used to perform a variety of database administration tasks including creating, starting, and shutting down databases, dumping and loading database files, importing and exporting data, and implementing database security.

- **PRO*Tools** — A set of utility programs to aid in developing and running applications in the OpenEdge Studio. For example, one of the PRO*Tools allows you to edit your PROPATH, the path OpenEdge searches to find programs. When you select PRO*Tools from the Tools menu, the OpenEdge Studio displays a resizeable palette of icons. You then run an individual PRO*Tool by clicking on its icon.

- **AppBuilder*** — The basic visual programming tool in the OpenEdge Studio. You fabricate and assemble application components in the AppBuilder. The Section Editor aids in editing ABL code conveniently while preserving the standardized structure that works best with OpenEdge application components, such as basic objects and SmartObjects.

- **Web tools** — Tools used to monitor your Web server environment, manipulate objects such as deleting and checking syntax, viewing database schema information, and issuing operating system commands through an intuitive browser interface that also provides links to sample applications and support services.

- **Results*** — An interactive menu-driven tool that lets end users with little or no programming experience query, report on, and maintain information stored in database tables. As an application developer, you can also customize Results according to the needs of individual sites, and then integrate it into your applications.

- **Translation Manager*** — A tool used to manage the process of customizing an application for specific end-user languages. Translation Manager allows you to select words and phrases from source procedure files for translation into other languages. These words and phrases are assembled into translation kits that you can send to translators who perform the actual translation. After you receive the completed translations, you use the Translation Manager to review the translations and integrate them into your applications. The Translation Manager also provides statistical information about your translation projects.
• **Visual Translator** — A tool used by the translators who translate the actual words and phrases into other languages. The Visual Translator uses a translation kit that was generated by the Translation Manager, and allows the translator to work either from a list or in the visual context of an application’s windows.

• **Application compiler** — A tool used to compile individual source procedures or a group of procedures to produce executable code that is permanently stored.

• **Application debugger** — An interactive tool used to monitor the execution of an application by stopping and restarting it at specified lines. You can also display or update information about the procedure you are running (such as the names and values of variables) without modifying any procedure code. The Debugger runs as a separate process with its own window.
OpenEdge online help system

OpenEdge provides an online help system that provides information about using the tools, reference material on ABL, Cue Cards for the SmartObjects, and explanatory information on OpenEdge error messages. The online help system contains information for both programming and system administration audiences.

The online help system uses hypertext links to access help topics. In addition to providing help information, help topics provide links to other help systems and information. Hypertext links appear on your screen as colored text with a solid underline. To go to the topic associated with the link, click the link text. After viewing the help, you can return to the previous screen by choosing the Back button.

You can access help information from the Desktop in the following ways:

- Press F1 to get context-sensitive help about the current window or dialog box. Most dialog boxes in the OpenEdge Studio also have a Help button that provides access to help information.

- Use the Help menu to access information about the current tool or the online reference guide. The Help system contains topics that describe what information you can access through each of the options on the Help menu.
Overview of the OpenEdge AppBuilder

This section provides an overview of the OpenEdge AppBuilder and includes information on starting the AppBuilder and a description of the AppBuilder windows, menu options, and the AppBuilder Object Palette. The AppBuilder is a visual programming environment that allows you to rapidly build complex applications. After you read this section, you should be familiar with the components of the AppBuilder and how to work with the AppBuilder and Object Palette options. For more detailed information about using the AppBuilder and Object Palette, see *OpenEdge Development: AppBuilder*.

**Note:** The components and menu options available in the AppBuilder differ depending upon the products you have licensed and the components that you choose to install. If you decide that you want access to a component that you did not initially install, you can add it to your installation. See *OpenEdge Getting Started: Installation and Configuration* for details.

To start the AppBuilder, choose the **AppBuilder** tool icon on the Desktop. The AppBuilder main window appears with its Object Palette, as shown in Figure 1–2.

![AppBuilder main window and Object Palette—initial display](image)

**Figure 1–2:** AppBuilder main window and Object Palette—initial display

**Note:** If your screen resolution is not standard SVGA (800 x 600 pixels), the appearance of the Object Palette might be different.

**AppBuilder main window**

When you start the AppBuilder, the main window and the Object Palette open. You can move these two windows independently on your screen. The Object Palette has several display options that do not depend on the **AppBuilder** main window. See the “**AppBuilder Object Palette**” section on page 1–11 for a discussion of these options.
The **AppBuilder** main window has four elements, as shown in Figure 1–3.

![AppBuilder main window](image)

**Figure 1–3:** The AppBuilder main window

The following sections describe the four elements of the **AppBuilder** main window.

### Menu bar

The menu bar provides access to most of the AppBuilder functions. Some menu options are not available when working in WebSpeed development mode:

- **File** — Provides access to file and window operations such as **New**, **Open**, **Close**, **Save**, and **Print**. You can also use this option to view and select from a list of most recently used files. The **Most Recently Used File List** option is enabled by default, and you can specify the number of files you want to appear in this list through the **Options → Preferences** menu item.

- **Edit** — Provides access to general editing operations such as copying, deleting, and pasting objects. In addition to these general operations, the **Edit** menu provides options for copying and inserting files, specifying the tab order for a frame, and accessing a specific page.

- **Compile** — Provides access to a number of options you can use to run, view, test, and debug your code.

- **Tools** — Provides access to a number of AppBuilder tools, such as the Data Dictionary and Procedure Editor, and a number of other options to help you design and build applications, such as a Color Editor and Property Sheets.

- **Options** — Provides access to AppBuilder preference options including colors and grid alignments.

- **Layout** — Provides access to options that you use to align objects using predefined layouts or customized layouts.

- **Window** — Provides access to the Section Editor, properties windows, and editors.

- **Help** — Provides access to help information, including error message explanations and Cue Cards for SmartObjects.

As you complete the tutorial exercises in this guide, you will become familiar with many of the menu selections. The menus and icons of the AppBuilder main window also provide access to all of the functions available in the Desktop, including the ability to start other tools and to access the online help system.

For more detailed information about the AppBuilder menu bar options, see the online help for that menu or the chapter on the AppBuilder interface in *OpenEdge Development: AppBuilder*. 
## Toolbar

Each icon in the AppBuilder toolbar provides quick access to frequently used menu options. Table 1–1 describes the function of each icon.

<table>
<thead>
<tr>
<th>Icon</th>
<th>ToolTip</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="New" /></td>
<td>New</td>
<td>Creates a new object file and displays an associated design window or tree view. The New icon on the toolbar corresponds to the File → New menu option.</td>
</tr>
<tr>
<td><img src="image2" alt="Open" /></td>
<td>Open</td>
<td>Opens an existing procedure file and displays its associated design window. The Open icon corresponds to the File → Open menu option.</td>
</tr>
<tr>
<td><img src="image3" alt="Save" /></td>
<td>Save</td>
<td>Saves the current procedure file. The Save icon corresponds to the File → Save menu option.</td>
</tr>
<tr>
<td><img src="image4" alt="Print" /></td>
<td>Print</td>
<td>Prints the currently selected procedure. When you select the Print icon, a Windows Print dialog box opens. From this Print dialog box, you can specify printing options and properties.</td>
</tr>
<tr>
<td><img src="image5" alt="Procedure Settings" /></td>
<td>Procedure Settings</td>
<td>Displays and allows editing of the procedure settings for the current procedure file. The Procedure Settings icon corresponds to the Tools → Procedure Settings menu option. Procedure settings maintain information about procedure files, the attributes they possess, and standardized ways in which they interact with other procedures.</td>
</tr>
<tr>
<td><img src="image6" alt="Run" /></td>
<td>Run</td>
<td>Compiles and runs the current procedure file. The Run icon corresponds to the Compile → Run menu option.</td>
</tr>
<tr>
<td><img src="image7" alt="Stop" /></td>
<td>Stop</td>
<td>Stops the currently running procedure. The Stop icon replaces the Run icon when you run a procedure.</td>
</tr>
<tr>
<td><img src="image8" alt="Edit Code" /></td>
<td>Edit Code</td>
<td>Opens the Section Editor, which enables you to edit certain sections of the procedure file. The Edit Code icon corresponds to the Window → Code Section Editor menu option.</td>
</tr>
<tr>
<td><img src="image9" alt="List Objects" /></td>
<td>List Objects</td>
<td>Opens the List Objects dialog box. This dialog box lists all of the objects currently in use in any design window. You can access an individual object’s property sheet by double-clicking an object in the list.</td>
</tr>
</tbody>
</table>
Current object fields

The current object fields display information about the current object, which is a single object in a design window that has design focus. If you select multiple objects in a design window, then no current object information is displayed.

The current object fields display the following information:

- **Object** — Specifies the name of the current object.

- **Title/Label/Text/Master/File/OCX** — For a basic object, this field specifies the object’s title or label, if it has one. For example, text objects have no label or title, so for a text object this field displays its contents.

  For SmartObject instances, which are based on prefabricated masters stored in separate files, this field displays the object’s title bar text (SmartWindows and SmartDialogs) or the pathname of the master file (all other SmartObjects).

  For an ActiveX (OCX), this field displays the name of the ActiveX object.

**Status area**

From left to right, the four fields in the status area display are:

- **File status** — Information about current AppBuilder processing

- **Page status** — The current page (applicable to SmartContainers only)

- **Object Palette status** — The type of object currently selected on the Object Palette

- **Lock status** — For inserting objects (see the “How the Object Palette responds to the mouse” section on page 1–15 for more information on lock status)
Overview of the OpenEdge AppBuilder

AppBuilder Object Palette

From the Object Palette, you can select basic objects, SmartObjects, and ActiveX (OCX) controls to insert into a design window. You can also create and insert new SmartObjects from the Object Palette.

Object Palette Options

The Object Palette’s menu options allow you to control some characteristics of the Object Palette itself. For example, when you choose \textbf{Menu} \rightarrow \textbf{Options} \rightarrow \textbf{Show Menu Only}, the icons are hidden and only the \textbf{Menu} options appear in the Object Palette. This option is useful when you are working with a low resolution and do not want to use valuable screen space to display the palette’s icons.

The other two Object Palette options give you control over other aspects of its appearance. If you choose \textbf{Menu} \rightarrow \textbf{Options} \rightarrow \textbf{Top-Only Window}, the Object Palette stays on top of all other windows, including design windows. If you choose \textbf{Menu} \rightarrow \textbf{Options} \rightarrow \textbf{Save Palette}, the AppBuilder saves the options, position, and size of the Object Palette for future sessions in the registry or in the \textit{progress.ini} file.

You can resize the Object Palette and move it to a more convenient area on your desktop or minimize it until you are ready to begin assembling your applications. To view an icon’s name, place the cursor over the icon to display the ToolTip.

The icons in the bottom rows of the Object Palette represent SmartObjects, as shown in Figure 1–4. With the exception of the Pointer, which represents no object, all other icons on the Object Palette represent basic objects, which are also referred to as widgets.

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{palette.png}
\caption{The AppBuilder Object Palette}
\end{figure}
Table 1–2 lists the icons on the Object Palette and their functions. For more information about the objects in the Object Palette, see OpenEdge Development: AppBuilder, OpenEdge Development: ADM and SmartObjects, and OpenEdge Development: ADM Reference.

### Table 1–2: Object Palette icons

<table>
<thead>
<tr>
<th>Icon</th>
<th>ToolTip</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="Image" alt="Pointer" /></td>
<td>Pointer</td>
<td>Selects an object in the design window or frame.</td>
</tr>
<tr>
<td><img src="Image" alt="DB-Fields" /></td>
<td>DB-Fields</td>
<td>Adds a database field and a query for that field.</td>
</tr>
<tr>
<td><img src="Image" alt="Query" /></td>
<td>Query</td>
<td>Adds a query and starts the QueryBuilder. The QueryBuilder widget is visible only at design time.</td>
</tr>
<tr>
<td><img src="Image" alt="Browse" /></td>
<td>Browse</td>
<td>Adds a browser.</td>
</tr>
<tr>
<td><img src="Image" alt="Frame" /></td>
<td>Frame</td>
<td>Adds a frame. Frames are used to visually and functionally group other objects.</td>
</tr>
<tr>
<td><img src="Image" alt="Rectangle" /></td>
<td>Rectangle</td>
<td>Adds a rectangle. Rectangles are used to visually group other objects.</td>
</tr>
<tr>
<td><img src="Image" alt="Image" /></td>
<td>Image</td>
<td>Adds a static image.</td>
</tr>
<tr>
<td><img src="Image" alt="Radio-Set" /></td>
<td>Radio-Set</td>
<td>Adds a radio button set. This object creates a three-button radio set with generic text.</td>
</tr>
<tr>
<td><img src="Image" alt="Toggle-Box" /></td>
<td>Toggle-Box</td>
<td>Adds a single check box with a generic label.</td>
</tr>
<tr>
<td><img src="Image" alt="Slider" /></td>
<td>Slider</td>
<td>Adds a slider. You can define sliders as horizontal or vertical.</td>
</tr>
<tr>
<td><img src="Image" alt="Button" /></td>
<td>Button</td>
<td>Adds a pushbutton with a generic label.</td>
</tr>
<tr>
<td><img src="Image" alt="Selection-List" /></td>
<td>Selection-List</td>
<td>Adds a scrollable selection list.</td>
</tr>
<tr>
<td><img src="Image" alt="Editor" /></td>
<td>Editor</td>
<td>Adds a multi-line text editor.</td>
</tr>
<tr>
<td><img src="Image" alt="Combo-Box" /></td>
<td>Combo-Box</td>
<td>Adds a non-OCX combo box.</td>
</tr>
<tr>
<td><img src="Image" alt="Fill-In" /></td>
<td>Fill-In</td>
<td>Adds a single-line editor.</td>
</tr>
<tr>
<td><img src="Image" alt="Text" /></td>
<td>Text</td>
<td>Adds text to a window. You can specify a font and color for the text.</td>
</tr>
<tr>
<td>Icon</td>
<td>ToolTip</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>---------</td>
<td>-------------</td>
</tr>
<tr>
<td>OCX</td>
<td>Adds an ActiveX control.</td>
<td></td>
</tr>
<tr>
<td>CSSpin</td>
<td>Adds a OpenEdge Spin Button control.</td>
<td></td>
</tr>
<tr>
<td>CSComboBox</td>
<td>Adds a OpenEdge OCX combo box.</td>
<td></td>
</tr>
<tr>
<td>PSTimer</td>
<td>Adds a OpenEdge OCX Timer object.</td>
<td></td>
</tr>
<tr>
<td>SmartDataObject</td>
<td>Adds a SmartDataObject (SDO) I/O interface. SmartDataObjects manage the data stream to and from disk, filter records, and respond to requests from other SmartObjects.</td>
<td></td>
</tr>
<tr>
<td>DataView</td>
<td>Adds a DataView. DataViews organize data from groups of temp-tables and ProDataSets.</td>
<td></td>
</tr>
<tr>
<td>SmartObject</td>
<td>Adds a generic ADM Smart template. You use this template to create new types of SmartObjects.</td>
<td></td>
</tr>
<tr>
<td>SmartFolder</td>
<td>Adds an ADM tabs widget. Tabs widgets are used in SmartFolders to provide a context in which you can visually and functionally divide an interface.</td>
<td></td>
</tr>
<tr>
<td>SmartPanel</td>
<td>Adds one of the predefined ADM button arrays. SmartPanels provide upstream control of other SmartObjects.</td>
<td></td>
</tr>
<tr>
<td>SmartDataBrowser</td>
<td>Adds an ADM data browser. SmartDataBrowsers provide a multi-record row/column interface to data.</td>
<td></td>
</tr>
<tr>
<td>SmartDataViewer</td>
<td>Adds an ADM data viewer. SmartDataViewers provide a single-record interface to data that you can arrange for a specific design.</td>
<td></td>
</tr>
<tr>
<td>SmartDataField</td>
<td>Adds customized capabilities to a SmartDataViewer.</td>
<td></td>
</tr>
<tr>
<td>SmartLOBField</td>
<td>Adds a SmartLOBField.</td>
<td></td>
</tr>
<tr>
<td>SmartFilter</td>
<td>Adds an ADM Smart Query-By-Form.</td>
<td></td>
</tr>
<tr>
<td>SmartToolBar</td>
<td>Adds an ADM menu and toolbar.</td>
<td></td>
</tr>
</tbody>
</table>
In addition to choosing an object from the Object Palette, you can choose an object from the Object Palette menu. When you choose an object from the menu, some options include specific properties for the object. For example, if you choose an Editor object from the menu, you have the additional option of choosing a Read-Only Editor object.

You can add OCX icons or submenu options to the Object Palette by choosing Menu→Add OCX and then choosing Add as Palette Icon or Add to Palette SubMenu. Specify the appropriate OCX file information. The icon or submenu option appears on the Object Palette.

**Customizing the Object Palette**

In addition to providing a variety of display options for the Object Palette, the AppBuilder allows you to customize its contents. Choose Menu→Use Custom to add to or modify the files that determine the Object Palette’s behavior and appearance.

For example, if you create new SmartObject templates, you can add them to the Object Palette. You can also change certain default settings for the standard SmartObject templates.

**Note:** Creating a new SmartObject template is different from fabricating a SmartObject master from an existing template, as you do in Chapter 4, “Overview of SmartObject Technology.”

<table>
<thead>
<tr>
<th>Icon</th>
<th>ToolTip</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="SmartContainer icon" /></td>
<td>Adds an outer-level container object, such as a SmartWindow, to another container, imposing a parent/child relationship on objects that do not normally inherit from each other.</td>
<td></td>
</tr>
<tr>
<td><img src="image" alt="SmartBusinessObject icon" /></td>
<td>Adds a special ADM SmartContainer that organizes the data streams of several SmartDataObjects.</td>
<td></td>
</tr>
<tr>
<td><img src="image" alt="SmartProducer icon" /></td>
<td>Adds an ADM message handler that creates messages to be transmitted by a Progress message service.</td>
<td></td>
</tr>
<tr>
<td><img src="image" alt="SmartConsumer icon" /></td>
<td>Adds an ADM message handler that receives messages transmitted by a Progress message service.</td>
<td></td>
</tr>
<tr>
<td><img src="image" alt="SmartRouter icon" /></td>
<td>Adds an ADM message handler that routes incoming messages from a SmartConsumer to appropriate SmartB2BOBObjects.</td>
<td></td>
</tr>
<tr>
<td><img src="image" alt="SmartB2BObject icon" /></td>
<td>Adds an ADM transformation object that transforms information to and from XML, based on a protocol shared between the message’s sender and recipient.</td>
<td></td>
</tr>
<tr>
<td><img src="image" alt="SmartSender icon" /></td>
<td>Adds a customizable ADM object that transforms information according to your rules, passing the result to a SmartProducer.</td>
<td></td>
</tr>
<tr>
<td><img src="image" alt="SmartReceiver icon" /></td>
<td>Adds a customizable ADM object that receives information from a SmartConsumer and then transforms it according to your rules.</td>
<td></td>
</tr>
</tbody>
</table>
How the Object Palette responds to the mouse

The icons on the Object Palette respond differently to the left and right mouse buttons:

- **Left mouse button** — When you click the left button, the AppBuilder immediately enters insertion mode for basic objects and the SmartFolder. For other SmartObjects, the AppBuilder displays a dialog box from which you can choose a SmartObject master. After you select a SmartObject, the AppBuilder enters insertion mode. When you choose the OCX icon, the AppBuilder displays a dialog box from which you choose an ActiveX control.

  You can lock an Object Palette icon to insert multiple instances of an object. To lock an icon for SmartObjects or ActiveX controls, click it once. If there is more than one object of this type available, select one from the Choose dialog box that opens. After the dialog box closes, a click the object’s icon again before inserting to lock it. To lock an icon for other objects on the palette, click twice on an object. A lock image appears over an icon when you lock it. Choose the Pointer object to remove the lock.

- **Right mouse button** — When you click the right button, the AppBuilder displays a pop-up menu for the object you chose. The options available on the pop-up menu depend on the selected object.

  For basic objects, the menu allows you to choose from among the available variations of the object; if there is only one type of that object available, the menu has only the Default option. This also applies to the SmartFolder.

  For SmartObjects, with the exception of the SmartFolder, the pop-up menu generally allows you to choose from among any available prefabricated SmartObject masters, or to create a new master from one of the SmartObject templates provided with the product. Instead of allowing you to create a new master, the pop-up menu for the SmartPanel allows you to create a new instance of one of the four SmartPanel templates.

  The pop-up menu for the SmartContainer has a Choose SmartContainer option. You use this option to select an existing container, such as a SmartWindow, SmartFrame, or SmartDialog. To create a new SmartContainer, choose the New icon on the AppBuilder toolbar and select the container you want from the New dialog box.

Advanced editing features in OpenEdge Studio

If your OpenEdge Studio installation runs under a development license, you have access to the following advanced editing features:

- **Color coding** — Also called color syntax-highlighting, color coding is a powerful feature in which the editor displays various types of ABL syntax elements such as keywords, comments, and character strings in distinct colors.

  For example, all keywords might appear in blue. Color coding makes it easy to understand the syntax and structure of a ABL program. You can customize the color coding used for language elements and/or change their style, for example, to apply a bold or italic typeface.

  Color coding is supported for the ABL and for many other programming and scripting languages, including HTML, C/C++, Perl, and Java.
Overview of the OpenEdge Studio

- **Embedded color coding** — Embedded color coding recognizes multiple languages in the same file and can display them with their own color-coding schemes. Using embedded color coding simplifies editing source files that contain code in multiple languages: the editor displays each language’s code using a distinct color-coding scheme and all other language-specific features.

  By default, the editors recognize only embedded SpeedScript, JavaScript, VBScript, and/or Java in HTML files.

- **Syntax expansion** — Syntax expansion occurs when you type an ABL keyword, statement, or block structure and then press the SPACEBAR. The editor expands the syntax of the keyword or other language element by inserting a template specific to it.

- **Alias expansion** — Alias expansion allows you to define shortcuts that extend or modify syntax expansion in the editors. You can use the aliases to insert more complete code and syntax templates when you type the shortcut and press the SPACEBAR.

  For example, when you type in **DISP**, then press the SPACEBAR, the editor does not expand this syntax by default. However, you can use alias expansion to define a code template to insert more of the statement’s syntax. Then when you type **DISP** and press the SPACEBAR, the editor expands to the template you defined.

  OpenEdge provides a number of default aliases. These aliases are in the file `p4gl.als` located in your working editor configuration directory. By default, this is the PROEDIT directory in your OpenEdge working directory. Whenever you use the Alias Editor dialog box to modify or add aliases for ABL source files, the editor saves the changes to the `p4gl.als` file.

- **Syntax indenting** — Syntax indenting in the editors occurs when you press the ENTER key at the end of a block statement. The editor inserts a new line and indents the cursor relative to the block statement.

  For example, if you create a ABL source file with the cursor positioned at the end of a line that contains the **DO** keyword, and you press the ENTER key, the editor inserts a new line and indents the cursor relative to the **DO** keyword.

- **SmartPaste** — The SmartPaste feature allows you to paste lines of text into a ABL source file and indent them relative to the surrounding code. This feature saves coding time by making it unnecessary to indent pasted code manually. It is especially useful when you copy and paste code among several source files.

  For example, if you select several lines of code in a source file, copy them to the clipboard using **CTRL-C**, then paste them inside a **FOR** block using **CTRL-V**, the editors use SmartPaste to indent the pasted lines correctly relative to the indentation level of the **FOR** loop.

For more information about the advanced editing features, see *OpenEdge Development: AppBuilder* and *Procedure Window and Common Dialog* help system.

Now that you are familiar with the Desktop and the AppBuilder main window, you are ready to learn more about developing applications using the tools and SmartObject technology. To help you learn, this guide introduces the SportsPro sample application, which was designed to highlight these features.
Working with the Sample Application

This chapter provides information on running and using the SportsPro sample application. This application is included when you perform a complete installation of the OpenEdge Studio. The sample application highlights useful programming techniques and solutions using SmartObjects and the Application Development Model (ADM) for a distribution business.

Note: The sample files are part of the Documentation and Samples CD that you received with your product. For more information, see the “Example procedures” section on page Preface–4.

In this chapter, you see the application’s Order Tracking System from an end user’s perspective. In later chapters, you will build its user interface (UI) and some of its SmartObject components. For more information about SmartObject technology, see Chapter 4, “Overview of SmartObject Technology.”

This chapter also briefly examines the other functions of the SportsPro sample application. When you are ready to look deeper into the sample application, follow the pointers provided in this chapter to learn why the application is coded the way it is.

This chapter includes the following sections:

- The SportsPro sample application
- Creating and connecting to the Sports2000 database
- Copying the sample files
- Compiling the sample files
- Running the sample application
- Working with the Order Tracking System
The SportsPro sample application

The SportsPro sample application was created to show how you can quickly build complex applications with SmartObjects. As it evolves, it demonstrates more techniques for effectively using OpenEdge products to solve a variety of business needs. The SportsPro sample application has something to interest you, no matter what environment you plan to develop applications for. You can run it as a stand-alone application, distribute it with an AppServer, or use the WebClient to run it across an Internet, intranet, or extranet connection.

As you become more familiar with the OpenEdge products, the sample application is a good place to explore for ideas on how to add functionality to your own applications. After you run the sample application, open its modules in the AppBuilder. Examining the objects in the AppBuilder and the Section Editor provides you with insights on using SmartObjects to build your own applications.
Creating and connecting to the Sports2000 database

Before running or assembling the sample application described in this chapter, you must create a copy of the Sports2000 database and connect to it.

To complete this task, you use the Data Dictionary tool. The Data Dictionary is part of the OpenEdge Studio. With it, you can create and connect to databases, as well as edit the definitions of tables, fields, indexes, and other database-related information.

Making a copy of the Sports2000 database allows you to make changes to the copy while preserving the original database unchanged for future use. You can also replace your working copy using the Replace If Exists option.

To create and connect to your copy of the Sports2000 database:

1. Choose Programs→OpenEdge→Desktop from the Windows Start menu.

2. Select the Data Dictionary icon from the Desktop.

3. When you run the Data Dictionary, it looks for a connected database. If there is no connected database, the Dictionary Startup dialog box appears:

4. Select Create a New Database and click OK. The default selection for the Dictionary Startup dialog box is Connect to an Existing Database. The Create Database dialog box appears:

5. Type <wrk>\databases\sports2000.db for the New Physical Database Name, where <wrk> is your OpenEdge working directory.

6. Select A Copy of Some Other Database in the Start with radio set. The default selection is An EMPTY Database.
7. Click the Files button, browse to the location of the original Sports2000 database, and select it.

**Note:** If you installed the sample files in the default location, the Sports2000 database is located in C:\Progress\OpenEdge.

8. Click OK.

The Data Dictionary creates a copy of the Sports2000 database in your working directory. The **Connect Database** dialog box appears and displays the name of the database you just created:

9. Click OK in the **Connect Database** dialog box. The Data Dictionary makes the connection and its startup window appears:

You have now created and connected to your working copy of the Sports2000 database. The Data Dictionary window shows the Sports2000 database and the tables in the database.
The icons in the toolbar govern what you can do with the Data Dictionary and what information displays for the selected database. When you choose an icon, buttons for varying tasks appear at the bottom of the window. **Table 2–1** lists the icons and their associated displays and tasks.

### Table 2–1: Data Dictionary icons

<table>
<thead>
<tr>
<th>Icon</th>
<th>Displays</th>
<th>Available task buttons</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Connected databases icon" /></td>
<td>Connected databases</td>
<td>Create Database Database Properties</td>
</tr>
<tr>
<td><img src="image" alt="Tables in selected database icon" /></td>
<td>Tables in selected database</td>
<td>Create Table Table Properties Delete Table</td>
</tr>
<tr>
<td><img src="image" alt="Sequences in selected database icon" /></td>
<td>Sequences in selected database</td>
<td>Create Sequence Sequence Properties Delete Sequence</td>
</tr>
<tr>
<td><img src="image" alt="Fields in selected table icon" /></td>
<td>Fields in selected table</td>
<td>Create Field Field Properties Delete Field</td>
</tr>
<tr>
<td><img src="image" alt="Indexes on selected table icon" /></td>
<td>Indexes on selected table</td>
<td>Create Index Index Properties Delete Index</td>
</tr>
</tbody>
</table>


11. When you are ready, close the Data Dictionary and return to the Desktop.
Copying the sample files

After creating a copy of the Sports2000 database and connecting to it, copy the tutorial sample files to your working directory and compile them. You can then experiment with the tutorial files while preserving the original files.

To copy the sample files:

1. Copy the files from `<install-dir>\src\sports2000\gui`, where `<install-dir>` is your OpenEdge install directory, to your OpenEdge working directory.

2. Create a new subfolder called **Images** under your OpenEdge working directory.

3. Copy the files from `<install-dir>\src\sports2000\images` to `<wrk>\Images`, where `<install-dir>` is your OpenEdge install directory and `<wrk>` is your OpenEdge working directory.
Compiling the sample files

To make your sample files run more efficiently, use the OpenEdge Application Compiler to compile the sample files you copied. The compiled files have the same name as the window, procedure, and include files, but have a .r file extension.

To compile the sample files:

1. Click the AppBuilder icon on the Desktop toolbar to start the AppBuilder. The AppBuilder main window appears:

2. Choose Tools→Application Compiler from the AppBuilder menu. The Application Compiler dialog box appears:

3. Click Start Compile. The Compiler Results dialog box appears showing the AppBuilder’s progress compiling the sample files.

   Note: You do not need to specify a path because the compiler automatically looks for files in your working directory.

4. When the compile finishes, click OK and exit the compiler.

You have created a copy of the Sports2000 database, connected to your database, copied the sample application files, and compiled them. You can now run the sample application.
Running the sample application

Running the SportsPro sample application gives you a feel for what you can build with the AppBuilder and SmartObject technology. The sample application contains several modules, each demonstrating a technique for solving a business problem. You will build parts of the Order Tracking System module in the tutorial and exercises later in this guide.

To run the SportsPro sample application:

1. If you have not already connected to the database, connect to your copy of the Sports2000 database. See “Creating and connecting to the Sports2000 database” section on page 2–3 for information about connecting to your copy of the Sports2000 database.

2. Select the AppBuilder icon from the Desktop toolbar to start the AppBuilder. One of the windows that appears by default is the PRO*Tools palette:

3. Click the Run button from the PRO*Tools palette. The Run Procedure dialog box appears:

4. Type wmain.w for the Name of Procedure to Run.

   If you copied the sample files as described in “Copying the sample files” section on page 2–6, you do not need to qualify the filename with its directory. The wmain.w file should be in your working directory which is a standard entry in your PROPATH.

5. Click Run. The main window of the SportsPro sample application appears:
Running the sample application

The SportsPro application’s main window is a SmartWindow. Most of the buttons on this SmartWindow use the FLAT–BUTTON attribute to create a two-dimensional look.

Table 2–2 lists the components used in the main window of the SportsPro sample application. It lists all the procedure files, all the SmartObjects, and any simple objects with important functions. Except for SmartObjects like SmartFilters for which the Application Development Model (ADM) supplies the masters, the procedure files and SmartObject master files listed in this table are in the gui folder in your working directory.

<table>
<thead>
<tr>
<th>Object or filename</th>
<th>Object type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>wmain.w</td>
<td>SmartWindow</td>
<td>Serves as a container for SDOs referenced by two of the modules and the buttons that launch the modules.</td>
</tr>
<tr>
<td>dcust</td>
<td>SmartDataObject (SDO)</td>
<td>Queries the Customer table to retrieve all customer records for the selected query. The query’s handle is passed into one of the modules.</td>
</tr>
<tr>
<td>BUTTON-1</td>
<td>Button</td>
<td>Launches wfndcust.w, passing it the handle for dcust.</td>
</tr>
<tr>
<td>BUTTON-2</td>
<td>Button</td>
<td>Launches wquery.w.</td>
</tr>
<tr>
<td>BUTTON-3</td>
<td>Button</td>
<td>Launches wdragdrop.w.</td>
</tr>
<tr>
<td>dorder</td>
<td>SDO</td>
<td>Queries the Order table to retrieve all order records for the selected query. The query’s handle is passed into one of the modules.</td>
</tr>
<tr>
<td>BUTTON-4</td>
<td>Button</td>
<td>Launches wfndord.w, passing it the handle for dorder.</td>
</tr>
<tr>
<td>BUTTON-5</td>
<td>Button</td>
<td>Launches wmntitem.w.</td>
</tr>
<tr>
<td>BUTTON-6</td>
<td>Button</td>
<td>Launches wordupd.w.</td>
</tr>
<tr>
<td>BUTTON-7</td>
<td>Button</td>
<td>Launches suppub.p.</td>
</tr>
<tr>
<td>BUTTON-8</td>
<td>Button</td>
<td>Launches gb2bintro.w. If you choose OK in the dialog box, it runs wstore.w.</td>
</tr>
<tr>
<td>BUTTON-9</td>
<td>Button</td>
<td>Launches worde.r.w.</td>
</tr>
</tbody>
</table>

Note: Table 2–2 and the tables in later sections present the objects and procedures from each module in an event-driven order. The objects contained in another object, and the procedures executed in that object, are listed directly after an object. Where data passes between objects and procedures, they are listed chronologically. Otherwise, objects are listed from top to bottom in their container.

Each of the buttons on the sample application’s main window launches a module. You can now examine these modules in turn. As you become more familiar with the OpenEdge products, you can return to these modules and examine them in greater detail. These modules show how to accomplish several common programming tasks.
Dynamic Query module

To add flexibility to your applications, ABL (Advanced Business Language) enables you to build dynamic queries. With the dynamic query structure, you do not have to anticipate and code every case for which an end user might want a query. At run time, end users can design their own queries to fit exactly the questions they need to answer.

To launch this module, choose the Dynamic Query button on the SportsPro main window. The Find Customer dialog box appears:

When you launch the Dynamic Query module, it accepts the handle of a SDO on the SportsPro main window to serve as the data source for the query. As you create a query using the buttons in the upper section of this window, the module records the contents of the For Each Customer Where editor as a variable. Choosing the Search button launches the module’s Search procedure. The Search procedure accepts the handle of the user input variable and uses it to replace the SDO’s WHERE and SORT clauses.
Table 2–3 describes the major objects used in the Dynamic Query module.

### Table 2–3: Components of the Dynamic Query module

<table>
<thead>
<tr>
<th>Object or filename</th>
<th>Object type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>wfindcust.w</td>
<td>SmartWindow</td>
<td>Serves as a container for the other objects in the module.</td>
</tr>
<tr>
<td>dcust</td>
<td>SDO</td>
<td>Queries the Customer table to retrieve all customer records for the selected query. When this module is launched, it accepts the handle for the instance of <code>dcust</code> in the SportsPro main window. <strong>Note:</strong> Because this module uses a handle passed in from the main window, you must run the main window and launch it from there.</td>
</tr>
<tr>
<td>getnewfields.p</td>
<td>Procedure file</td>
<td>Runs during the <code>initializeObject</code> internal procedure to populate the <code>lcustfieldname</code> selection list.</td>
</tr>
<tr>
<td>lcustfieldname</td>
<td>Selection-list</td>
<td>Lists the fields in the Customer table that can be added to the query.</td>
</tr>
<tr>
<td>qrystringdisp</td>
<td>Editor</td>
<td>Displays the query being built. Its contents are passed to the <code>Search</code> internal procedure.</td>
</tr>
<tr>
<td>btnsearch</td>
<td>Button</td>
<td>Runs the <code>Search</code> internal procedure to display the records matching the query shown in the <code>qrystringdisp</code> editor.</td>
</tr>
<tr>
<td>sortfld</td>
<td>Combo-Box</td>
<td>Lists the fields by which the result set can be sorted.</td>
</tr>
<tr>
<td>lsort</td>
<td>Radio-set</td>
<td>Sets whether the result set is sorted in ascending or descending order.</td>
</tr>
<tr>
<td>bcust</td>
<td>SDB</td>
<td>Enables the user to select a customer record based on the query shown in the editor and the sort criteria.</td>
</tr>
<tr>
<td>btndspcust</td>
<td>Button</td>
<td>Launces <code>wdspcust.w</code>.</td>
</tr>
<tr>
<td>wdspcust.w</td>
<td>SmartWindow</td>
<td>Serves as a container for the other objects in the Display Detail section of the module.</td>
</tr>
<tr>
<td>vcust2</td>
<td>SmartDataViewer (SDV)</td>
<td>Displays full details of the customer currently selected in <code>bcust</code>.</td>
</tr>
<tr>
<td>stxtfld</td>
<td>SDF</td>
<td>Displays the Comments field in <code>vcust2</code> in a Cut/Copy/Paste editor.</td>
</tr>
</tbody>
</table>

**Note:** The Dynamic Query module also runs when you choose the **Find Customer** button in the **Customer** section of the SportsPro Order Tracking System.
For more information on building dynamic queries, see the chapter on using dynamic queries in *OpenEdge Deployment: Managing ABL Applications*.

**Dynamic Buffer/Temp Table module**

ABL puts the records that it reads into a buffer. Once a record is in the buffer, its fields are available to your application. While you can assign a static temp table for that buffer at compile time, you could not then change it at run time. For greater flexibility, ABL enables you to build dynamic buffers. With a dynamic buffer, you can select the table for your buffer at run time. The buffer fields enable you to select the fields to be displayed.

To launch this module, choose the **Dynamic Buffer/Temp Table** button on the **SportsPro** main window. The **Dynamic Buffer** dialog box appears:

![Dynamic Buffer/Temp Table dialog box]

This module demonstrates a non-SmartObject approach to separating the user interface and business logic portions of an application. The initializeObject procedure runs a procedure on the AppServer partition that populates the list of database tables. Choosing the **Open Query** button runs another procedure on the AppServer partition. That procedure populates a dynamic temp table and passes back the handle for that temp table.
Table 2–4 describes the major objects used in the Dynamic Buffer/Temp Table module.

Table 2-4: Components of the Dynamic Buffer/Temp Table module

<table>
<thead>
<tr>
<th>Object or filename</th>
<th>Object type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>wquery.w</td>
<td>SmartWindow</td>
<td>Serves as a container for the other objects in the module</td>
</tr>
<tr>
<td>gettablefields.p</td>
<td>Procedure file</td>
<td>Runs during the initializeObject internal procedure to populate the selecttable selection list</td>
</tr>
<tr>
<td>selecttable</td>
<td>Selection-list</td>
<td>Lists the available tables</td>
</tr>
<tr>
<td>getnewfields.p</td>
<td>Procedure file</td>
<td>Runs when a table is selected to populate the selectfld selection list</td>
</tr>
<tr>
<td>selectfld</td>
<td>Selection-list</td>
<td>Lists the available fields in the selected table</td>
</tr>
<tr>
<td>btnadd</td>
<td>Button</td>
<td>Adds the field selected in the selectfld selection list to the selectfld2 selection list through the additem internal procedure</td>
</tr>
<tr>
<td>btnremove</td>
<td>Button</td>
<td>Removes the selected field from the selectfld2 selection list through the removeitem internal procedure</td>
</tr>
<tr>
<td>selectfld2</td>
<td>Selection-list</td>
<td>Lists the fields that will be displayed when the query runs</td>
</tr>
<tr>
<td>btnopenqry</td>
<td>Button</td>
<td>Runs getrecords.p and receives the handle of the dynamic temp-table that procedure creates; then it runs the displayflds internal procedure to populate EDITOR–1</td>
</tr>
<tr>
<td>getrecords.p</td>
<td>Procedure file</td>
<td>Creates a dynamic temp-table and populates it based on the fields listed in the selectfld2 selection list</td>
</tr>
<tr>
<td>EDITOR–1</td>
<td>Editor</td>
<td>Displays a single record from the dynamic temp-table</td>
</tr>
<tr>
<td>btnfirst</td>
<td>Button</td>
<td>Navigates to the first row of the result set</td>
</tr>
<tr>
<td>btnnext</td>
<td>Button</td>
<td>Navigates to the next row of the result set</td>
</tr>
<tr>
<td>btnprev</td>
<td>Button</td>
<td>Navigates to the previous row of the result set</td>
</tr>
<tr>
<td>btnlast</td>
<td>Button</td>
<td>Navigates to the last row of the result set</td>
</tr>
</tbody>
</table>

For more information on building dynamic buffers and temp tables, see the chapters on using dynamic buffers and on using temp tables in *OpenEdge Getting Started: ABL Essentials.*
Drag and Drop module

ABL can read the names of files that an end user drops onto certain widgets. With this functionality, the end user can populate widgets, like selection lists, with a list of files. ABL can also use the filenames to retrieve the files and open them in appropriate widgets. Using these techniques, you could save your users time by allowing them to load data into your program by dragging a filename from the Windows Explorer or Desktop onto a widget in your application.

To launch this module, choose the Drag and Drop button on the SportsPro main window. The Drag and Drop dialog box appears:

Implementing this technique is simple. You set the DROP–TARGET attribute on the property sheet of a selection list or editor widget and code the trigger event. For images, the DROP–TARGET attribute is set on the frame that holds the image. When a file is dropped on an object with the DROP–TARGET attribute enabled, the DROP–FILE–NOTIFY event fires.

Note: If you cannot click on an object, it is probably behind another object in the window’s layout. Use the Layout menu to move the top object back, or use the List Objects dialog box to access the bottom object.
Table 2–5 describes the major objects used in the Drag and Drop module.

Table 2–5: Components of the Drag and Drop module

<table>
<thead>
<tr>
<th>Object or filename</th>
<th>Object type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>wdragdrop.w</td>
<td>SmartWindow</td>
<td>Serves as a container for the other objects in the module</td>
</tr>
<tr>
<td>fMain</td>
<td>SmartFrame</td>
<td>Acts as the drop-target for image files; it reads the filename and then displays the image.</td>
</tr>
<tr>
<td>SELECT–1</td>
<td>Selection-list</td>
<td>Acts as the drop-target for filenames; it reads the filenames and then adds them to the list of selections.</td>
</tr>
<tr>
<td>EDITOR–1</td>
<td>Editor</td>
<td>Acts as the drop-target for text files; it reads the filename and then opens the file in the editor.</td>
</tr>
</tbody>
</table>

For more information on how drag and drop works, see the entries for the DROP–TARGET attribute, the END–FILE–DROP( ) method, the GET–DROPPED–FILE( ) method, the DROP–FILE–NOTIFY event, and the NUM–DROPPED–FILES attribute in OpenEdge Deployment: Managing ABL Applications.

Resizable Dynamic Browser

ABL supports end users changing the way the data is displayed in a data browser. The changes occur only in the user interface; they do not change field widths or column positions in the data source.
To launch this module, choose the **Resizable Dynamic Browser** button on the **SportsPro** main window. The **Find Order** dialog box appears:

In a normal browser, these attributes are set on the **Advanced Properties** dialog box in the AppBuilder at design time. Because the browser is dynamic, the attributes have to be set programmatically. The browser also uses the dynamic filters supplied as part of the ADM.

Table 2–6 describes the major objects used in the Resizable Dynamic Browser module.

**Table 2–6: Components of the Resizable Dynamic Browser module (1 of 2)**

<table>
<thead>
<tr>
<th>Object or filename</th>
<th>Object type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>wfindord.w</td>
<td>SmartWindow</td>
<td>Serves as a container for the other objects in the module.</td>
</tr>
<tr>
<td>dorder</td>
<td>SDO</td>
<td>Queries the Order table to retrieve all order records for the query. The module accepts the handle for the instance of dorder in the SportsPro main window. <strong>Note:</strong> Because this module uses a handle passed in from the main window, you must run the main window and launch it from there.</td>
</tr>
<tr>
<td>getnewfields.p</td>
<td>Procedure file</td>
<td>Runs during the initializeObject internal procedure to populate the seldspflds selection list.</td>
</tr>
<tr>
<td>dynfilter</td>
<td>SmartFilter</td>
<td>Enables the user to dynamically filter the records shown in the browser.</td>
</tr>
</tbody>
</table>
Table 2–6: Components of the Resizable Dynamic Browser module (2 of 2)

<table>
<thead>
<tr>
<th>Object or filename</th>
<th>Object type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>seldspflds</td>
<td>Selection-list</td>
<td>Lists the fields in the Order table that can be displayed in the browser.</td>
</tr>
<tr>
<td>BUTTON-2</td>
<td>Button</td>
<td>Creates a dynamic browser instance from the master file, adm\2\dynbrowser.\w, using information from the filter and selection list.</td>
</tr>
<tr>
<td>dynbrowser</td>
<td>SmartDataBrowser (SDB)</td>
<td>Enables the user to browse a result set created dynamically from dorder and information from the filter and selection list.</td>
</tr>
<tr>
<td>tglbrowserresizable</td>
<td>Toggle-box</td>
<td>Sets the RESIZABLE and SELECTABLE attributes of the browser.</td>
</tr>
<tr>
<td>tglcolmovable</td>
<td>Toggle-box</td>
<td>Sets the COLUMN–MOVABLE attribute of the browser.</td>
</tr>
<tr>
<td>tglcolumnnresizable</td>
<td>Toggle-box</td>
<td>Sets the COLUMN–RESIZABLE attribute of the browser.</td>
</tr>
<tr>
<td>tglrowresize</td>
<td>Toggle-box</td>
<td>Sets the ROW–RESIZABLE attribute of the browser.</td>
</tr>
</tbody>
</table>

SmartDataObject module

One of the SmartObjects that OpenEdge uses to support distributed applications is the SmartDataObject (SDO). The SDO manages the query, validation, and transaction logic in one object. One of its important features is that it can run locally, or be split between the client and AppServer in a distributed environment. Separating user interface from business logic is one of the keys to creating distributable applications. The SDO also provides an open interface that you can access in an Open Client environment as a set of non-ABL objects.
To launch this module, choose the SmartDataObject button on the SportsPro main window. The Item Maintenance dialog box appears:

The SDO in this module queries the database and sends the records to a SDB and a SDV. The SDB and SDV are thin objects. They display data and accept changes to fields, but do not contain data update or validation logic. The SDO carries out the update and validation tasks on the server side.

Table 2–7 describes the major objects used in the SmartDataObject module.

Table 2–7: Components of the SmartDataObject module

<table>
<thead>
<tr>
<th>Object or filename</th>
<th>Object type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>wmntitem.w</td>
<td>SmartWindow</td>
<td>Serves as a container for the other objects in the module</td>
</tr>
<tr>
<td>ditem</td>
<td>SDO</td>
<td>Queries the Item table to retrieve item records</td>
</tr>
<tr>
<td>getcategory.p</td>
<td>Procedure file</td>
<td>Runs during the initializeObject internal procedure to populate the seldspfdls selection list</td>
</tr>
<tr>
<td>bitem</td>
<td>SDB</td>
<td>Enables the user to select an item record</td>
</tr>
<tr>
<td>lcategory1</td>
<td>Combo-box</td>
<td>Filters the items shown in bitem to those matching the selected category</td>
</tr>
<tr>
<td>vitem</td>
<td>SDV</td>
<td>Displays detailed information for the item currently selected in bitem</td>
</tr>
</tbody>
</table>
Running the sample application

Note: The SmartDataObject module also runs when you choose the View→Item Maintenance menu option in the SportsPro Order Tracking System.

For more information on SDOs, see the chapter on SmartObjects in OpenEdge Development: ADM and SmartObjects.

SmartDataField module

The SmartDataField (SDF) gives you a method to customize how particular pieces of data are displayed. You might use the SDF as a calculator to update an integer field, or as an editor with cut, copy, and paste buttons to update a character field. In this module, a SDF displays a date field as a calendar. Choosing a new date in the calendar changes the value in the field.

To launch this module, click the SmartDataField button on the SportsPro main window. The Order Update dialog box appears:

SDFs work with SDVs. You create a master for the SDF that you want to use. Then you drop an instance of it onto an appropriate field in a SDV to create the necessary links.
Table 2–8 describes the major objects used in the SmartDataField module.

Table 2–8: Components of the SmartDataField module

<table>
<thead>
<tr>
<th>Object or filename</th>
<th>Object type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>wordupd.w</td>
<td>SmartWindow</td>
<td>Serves as a container for the other objects in the module.</td>
</tr>
<tr>
<td>dorder</td>
<td>SDO</td>
<td>Queries the Order table to retrieve all order records.</td>
</tr>
<tr>
<td>vorder2</td>
<td>SDV</td>
<td>Displays the contents of several fields of the selected order record.</td>
</tr>
<tr>
<td>sdfcal</td>
<td>SDF</td>
<td>Displays the value of the OrderDate, PromiseDate, and ShipDate fields in the selected record. The SDFs also provide a more convenient method for changing the values than a standard fill-in field.</td>
</tr>
<tr>
<td>pnavico</td>
<td>SmartPanel (navigation)</td>
<td>Controls navigation for the result set from dorder.</td>
</tr>
<tr>
<td>pupdsav</td>
<td>SmartPanel (update)</td>
<td>Controls record update functions on the result set from dorder.</td>
</tr>
</tbody>
</table>

For more information on SDFs, see the chapter on SmartObjects in *OpenEdge Development: ADM and SmartObjects*.

**Publish and Subscribe module**

The Publish and Subscribe module demonstrates the ABL named-event functionality. The module uses two event publishers, publisher1.w and publisher2.w, and two event subscribers, subscriber1.w and subscriber2.w. A driver procedure, subpub.p, starts the publisher procedures as persistent procedures. The driver then starts the subscriber procedures, passing to them the handles for the persistent publisher procedures.
To launch this module, choose the **Publish and Subscribe** button on the **SportsPro** main window. The following dialog boxes appear:

The publish and subscribe functionality works like an e-mail alias. When users subscribe to an e-mail alias, they receive messages whenever messages are published to that alias. Other persistent procedures running in the same session can locate a procedure that generates an event of interest. The persistent procedures subscribe to that event by mapping the event to an internal procedure that is executed when the event occurs.

Table 2–9 describes the major objects used in the Publish and Subscribe module.

### Table 2–9: Components of the Publish and Subscribe module

<table>
<thead>
<tr>
<th>Object or filename</th>
<th>Object type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>suppup.p</td>
<td>Procedure file</td>
<td>Runs the four windows as persistent procedures and passes the handles for the publishers to the subscribers.</td>
</tr>
<tr>
<td>publisher1.w</td>
<td>SmartWindow</td>
<td>Serves as a container for the objects in it.</td>
</tr>
<tr>
<td>BUTTON–1</td>
<td>Button</td>
<td>Publishes a named event, “EventOne,” which causes the EventOne internal procedure to run in any subscriber for which it currently has a handle. It reports the number of subscribers to the event from the replies it receives.</td>
</tr>
<tr>
<td>BUTTON–2</td>
<td>Button</td>
<td>Publishes a named event, “EventTwo,” which causes the EventTwo internal procedure to run in any subscriber for which it currently has a handle.</td>
</tr>
<tr>
<td>publisher2.w</td>
<td>SmartWindow</td>
<td>Serves as a container for the objects in it.</td>
</tr>
</tbody>
</table>
For more information on publishing and subscribing to named events, see the chapter on user-defined functions in *OpenEdge Getting Started: ABL Essentials.*

### SmartB2B and SonicMQ module

This module demonstrates the use of several new SmartObjects to link different businesses using a “Publish and Subscribe” technique. You might also do this using a “Point-to-Point” technique. A store can transmit its orders to its supplier as XML documents in a previously agreed upon format. The supplier can receive the XML document and translate the data into a form compatible with its own applications.

<table>
<thead>
<tr>
<th>Object or filename</th>
<th>Object type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>BUTTON–1</td>
<td>Button</td>
<td>Publishes a named event, “EventOne,” which causes the EventOne internal procedure to run in any subscriber for which it currently has a handle. It reports the number of subscribers to the event from the replies it receives.</td>
</tr>
<tr>
<td>BUTTON–2</td>
<td>Button</td>
<td>Publishes a named event, “EventTwo,” which causes the EventTwo internal procedure to run in any subscriber for which it currently has a handle.</td>
</tr>
<tr>
<td>subscriber1.w</td>
<td>SmartWindow</td>
<td>Serves as a container for the objects in it.</td>
</tr>
<tr>
<td>BUTTON–3</td>
<td>Button</td>
<td>Subscribes to both named events in publisher1.w and to “EventOne” in publisher2.w.</td>
</tr>
<tr>
<td>BUTTON–4</td>
<td>Button</td>
<td>Unsubscribes from “EventOne” in publisher1.w.</td>
</tr>
<tr>
<td>BUTTON–5</td>
<td>Button</td>
<td>Unsubscribes from “EventOne” in both publishers.</td>
</tr>
<tr>
<td>subscriber2.w</td>
<td>SmartWindow</td>
<td>Serves as a container for the objects in it.</td>
</tr>
<tr>
<td>BUTTON–6</td>
<td>Button</td>
<td>Subscribes to “EventTwo” in both publishers and to “EventOne” in publisher1.w.</td>
</tr>
</tbody>
</table>
To launch this module:

1. Click the **SmartB2B and SonicMQ** button on the **SportsPro** main window. A window appears explaining the environment that you need to set up to run this module.

2. Follow the instructions to launch the first application in this module. The **Send Purchase Order** dialog box appears:

![Send Purchase Order Dialog Box](image)

3. Follow the instructions to launch the second application in this module. The **Order Fulfillment** dialog box appears:

![Order Fulfillment Dialog Box](image)

Each of the applications that make up this module contains a SmartBusinessObject (SBO) that organizes the data streams of several SDOs. A SmartB2BObject takes the data from the SBO and runs it through a mapping file generated by the OpenEdge XML Mapping Tool. The XML message is then sent out by a SmartProducer and transmitted using SonicMQ. The file is received at the other end by a SmartConsumer and passed to its SmartB2BObject to be decoded with another XML mapping file.
Table 2–10 describes the major objects used in the SmartB2B and SonicMQ module.

<table>
<thead>
<tr>
<th>Object or filename</th>
<th>Object type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>gb2bintro.w</td>
<td>SmartDialog</td>
<td>Introduces the requirements for running this module. If you choose OK, wstore.w runs.</td>
</tr>
<tr>
<td>wstore.w</td>
<td>SmartWindow</td>
<td>Serves as a container for the other objects in the retailer’s application.</td>
</tr>
<tr>
<td>sbopo</td>
<td>SmartBusinessObject</td>
<td>Organizes the data streams of dcust, ditem, dpo, and dpoline into a single logical data stream.</td>
</tr>
<tr>
<td>dcust</td>
<td>SDO</td>
<td>Queries the Customer table to retrieve all customer records, and supplies them to sbopo.</td>
</tr>
<tr>
<td>ditem</td>
<td>SDO</td>
<td>Queries the Item table to retrieve all item records, and supplies them to sbopo.</td>
</tr>
<tr>
<td>dpo</td>
<td>SDO</td>
<td>Queries the PurchaseOrder table to retrieve all purchase order records, and supplies them to sbopo.</td>
</tr>
<tr>
<td>dpoline</td>
<td>SDO</td>
<td>Queries the POLine table to retrieve all purchase order line records, and supplies them to sbopo.</td>
</tr>
<tr>
<td>vpo</td>
<td>SDV</td>
<td>Displays fields from sbopo.</td>
</tr>
<tr>
<td>dynbrowser</td>
<td>SDB</td>
<td>Displays details of the line items in the record selected from sbopo.</td>
</tr>
<tr>
<td>pnavico</td>
<td>SmartPanel (navigation)</td>
<td>Controls navigation within the result set from sbopo.</td>
</tr>
<tr>
<td>BUTTON–1</td>
<td>Button</td>
<td>Runs sendMessage( ) in sbbpo.</td>
</tr>
<tr>
<td>sbbpo</td>
<td>SmartB2BOBJECT</td>
<td>Using the predetermined schema, creates an XML message containing the data from the current record and passes it to dynproducer.</td>
</tr>
<tr>
<td>dynproducer</td>
<td>SmartProducer</td>
<td>Sends the XML message to the SonicMQ broker for delivery to the remote procedure.</td>
</tr>
<tr>
<td>wsupplier.w</td>
<td>SmartWindow</td>
<td>Serves as a container for the other objects in the supplier’s application.</td>
</tr>
<tr>
<td>Note: Since this module demonstrates how separate applications can exchange data, you have to start this application in a separate session.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>dynconsumer</td>
<td>SmartConsumer</td>
<td>Receives the XML message from the SonicMQ broker and passes it to sbborder.</td>
</tr>
<tr>
<td>sbborder</td>
<td>SmartB2BOBJECT</td>
<td>Reads the data from the XML message and uses the predetermined schema to transform the XML message into a record in sboord.</td>
</tr>
</tbody>
</table>
Running the sample application

For more information on using SmartB2B objects and integrating OpenEdge applications with SonicMQ, see the chapter on SmartObjects in *OpenEdge Development: ADM and SmartObjects* and the chapter on accessing SonicMQ messaging from the ABL in *OpenEdge Development: Messaging and ESB*.

The SportsPro sample application modules that you examined are useful starting points. In them, you can see how to make better use of OpenEdge’s capabilities in your applications. To see how to integrate separate modules into a functioning application, you can now look at the last part of the sample application, the Order Tracking System. The Order Tracking System integrates most of the modules you have just examined and several others into a finished application.

### Table 2–10: Components of the SmartB2B and SonicMQ module (2 of 2)

<table>
<thead>
<tr>
<th>Object or filename</th>
<th>Object type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>sboord</td>
<td>SmartBusinessObject</td>
<td>Organizes the data streams of dord and dordline into a single logical data stream.</td>
</tr>
<tr>
<td>dord</td>
<td>SDO</td>
<td>Queries the Order table to retrieve all order records, and supplies them to sboord.</td>
</tr>
<tr>
<td>dordline</td>
<td>SDO</td>
<td>Queries the OrderLine table to retrieve all order line records, and supplies them to sboord.</td>
</tr>
<tr>
<td>vord</td>
<td>SDV</td>
<td>Displays fields from sboord.</td>
</tr>
<tr>
<td>dynbrowser</td>
<td>SDB</td>
<td>Displays details of the line items in the record selected from sboord.</td>
</tr>
<tr>
<td>pnavico-2</td>
<td>SmartPanel (navigation)</td>
<td>Controls navigation within the result set from sboord.</td>
</tr>
</tbody>
</table>
Working with the Sample Application

Working with the Order Tracking System

The Order Tracking System combines modules to answer the common business functions needed in an order tracking application. Using it, you can add, search for, and maintain particular customers and orders. You can also maintain the product catalog and warehouse inventory records.

Business scenario

The architecture of the SportsPro Order Tracking System reflects a possible solution to a realistic business problem. Imagine that you are the general manager of a growing sporting goods distribution company. You find that your sales people are spending too much time responding to customers’ inquiries about the status of existing orders and too little time generating new orders. Your solution is to institute an order tracking help desk to enter the new orders for the sales people and to field customer inquiries.

You start designing an application for users to quickly look up customer information. You want the application structured so the tasks that users need do most often are the easiest to accomplish. You determine that users most often need to:

- Enter new orders
- Search for a customer
- Enter a new customer
- Change customer information, such as an address or phone number
- Check inventory quantities
- Search for an order
- Check on the shipping status of an order as a whole
- Retrieve information about specific items that were ordered
- Modify lines in an order

After you know what tasks users will do with the application, you know what information they will need. From that, you create a database that contains the information you need to query for the tasks. Once you have your database, you can begin the development phases.
To begin examining the Order Tracking System, click the **Order Tracking** button from the **SportsPro** main window. The main window of the Order Tracking System appears:

![Order Tracking System](image)

**Note:** You might encounter display problems with the application screens if you have your display set to large fonts and a resolution of 800x600. As a rule, if you use large or small fonts, you should use a higher resolution setting, such as 1024x768. If you do encounter display problems with the application screens, try changing your display settings.

The main window is a SmartWindow that is divided into three parts:

- Customer section
- Order section
- Order Line section

Table 2–11 describes the major objects used in the main window of the Order Tracking System.

### Table 2–11: Components of the Order Tracking System main window (1 of 3)

<table>
<thead>
<tr>
<th>Object or filename</th>
<th>Object type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>worder.w</td>
<td>SmartWindow</td>
<td>Serves as a container for the other objects in the module</td>
</tr>
<tr>
<td>m_Item_Maintenance</td>
<td>Menu-item</td>
<td>Launches <code>wmntitem.w</code>, which is described in Table 2–7</td>
</tr>
</tbody>
</table>
### Working with the Sample Application

**Table 2–11: Components of the Order Tracking System main window (2 of 3)**

<table>
<thead>
<tr>
<th>Object or filename</th>
<th>Object type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>m_Inventory_Trans</td>
<td>Menu-item</td>
<td>Launches <code>winvtrans.w</code> (the Inventory Transfer section is described in Table 2–15)</td>
</tr>
<tr>
<td>m_Set_Application _Language</td>
<td>Menu-item</td>
<td>Launches <code>wlang.w</code> (the Set Application Language section is described in Table 2–16)</td>
</tr>
<tr>
<td>dcust</td>
<td>SDO</td>
<td>Queries the Customer table to retrieve all customer records for the selected query</td>
</tr>
<tr>
<td>vcust</td>
<td>SDV</td>
<td>Displays the contents of several fields of the selected customer record</td>
</tr>
<tr>
<td>bfindcust</td>
<td>Button</td>
<td>Launches <code>wfindcust.w</code>, passing it the handle of <code>dcust</code> (<code>wfindcust.w</code> is described in Table 2–3)</td>
</tr>
<tr>
<td>bmntcust</td>
<td>Button</td>
<td>Launches <code>wmntcust.w</code>, passing it the handle of <code>dcust</code> (the Customer Maintenance section is described in Table 2–12)</td>
</tr>
<tr>
<td>dorder</td>
<td>SDO</td>
<td>Queries the Order table to retrieve all order records for the selected query</td>
</tr>
<tr>
<td>dordline</td>
<td>SDO</td>
<td>Queries the OrderLine table to retrieve all order line records for the selected query</td>
</tr>
<tr>
<td>vorder</td>
<td>SDV</td>
<td>Displays the contents of several fields from the Order and Salesrep tables for the selected order record</td>
</tr>
<tr>
<td>dsalesrep</td>
<td>SDO</td>
<td>Queries the Salesrep table to retrieve all sales rep records for the selected query</td>
</tr>
<tr>
<td>dynselect</td>
<td>SmartSelect</td>
<td>Allows the user to navigate through the records in <code>dsalesrep</code></td>
</tr>
<tr>
<td>CtrlFrame</td>
<td>OCX</td>
<td>Provides navigation control for the order records</td>
</tr>
<tr>
<td>CtrlFrame-2</td>
<td>OCX</td>
<td>Provides navigation control for the order records</td>
</tr>
<tr>
<td>bcreord</td>
<td>Button</td>
<td>Launches <code>wmntord.w</code>, passing it the handles of <code>dcust</code>, <code>dorder</code>, and <code>dordline</code>. (the Order Entry/Maintenance section is described in Table 2–13)</td>
</tr>
<tr>
<td>bfindord</td>
<td>Button</td>
<td>Launches <code>wfindord2.w</code>, passing it the handle of <code>dorder</code> (the Find Order section is described in Table 2–14)</td>
</tr>
</tbody>
</table>
Customer section

The Customer section of the Order Tracking System main window has an SDV that displays customer information. The Customer section has the following buttons:

- **Find Customer** — Opens a dialog box to search for customer information
- **Maintain Customer** — Opens a dialog box to modify customer information

Searching for customers

The **Find Customer** button in the Order Tracking System main window opens the **Find Customer** window. In this window, you can view information about all customers, or you can find customers and view information based on specific search criteria.

The **Find Customer** window is a nonmodal child window of the main window. As a result, you can move between the **Find Customer** window and the main window by clicking on either one.

You can search the Sports2000 database dynamically by specifying query criteria at run time. Customers selected by the dynamic query appear in the browser.
To search for a customer:

1. Click the **Find Customer** button. The **Find Customer** window appears:

2. Using the **Customer** selection list and buttons in the upper section of the window, build the following query in the editor:

   **Name BEGINS "B"**

3. Click the **Search** button.

   The SDB displays customer names that start with “B”. From this window, you view details about a selected customer. You can also use the **Show All** button to remove the query and to view all the customers.
4. Select a row in the browser and choose the **Display Detail** button. The **Display Customer** dialog box appears:

![Display Customer dialog box](image)

The **Display Customer** dialog is a child window of the **Find Customer** window. The **Display Customer** dialog has a SDV for displaying address and other information for the customer selected in the **Find Customer** window. To learn how to create this SDV, see Chapter 4, “Overview of SmartObject Technology.”

5. After viewing the information, close the windows and return to the **Order Tracking System** main window.

**Note:** Table 2–3 describes the important elements of these windows in detail.

### Modifying customer information

The **Maintain Customer** button in the Order Tracking System main window opens the **Maintain Customer** window. This window uses an instance of the same SDV that you saw in the **Find Customer** window. In this instance, the SDV is locally configured to allow you to change the record. This window also uses a SDF on the Comments field to enable you to view and enter comments. There is also an instance of a dynamic SmartToolBar that you use to navigate, create, and maintain customer records.

The **Maintain Customer** window is a nonmodal child window of the main window. As a result, you can move between this window and the main window by clicking on either one.
To modify customer information:

1. Click the **Maintain Customer** button. The **Maintain Customer** window appears:

   ![Maintain Customer Window](image)

2. Experiment with entering and updating changes in a customer’s address, telephone number, contact, e-mail, and fax information.

3. Close the window and return to the main window.

Table 2–12 describes the major objects in the Maintain Customer section of the Order Tracking System.

<table>
<thead>
<tr>
<th>Object or filename</th>
<th>Object type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>wmntcust.w</td>
<td>SmartWindow</td>
<td>Serves as a container for the objects in the Customer Maintenance section of this module.</td>
</tr>
<tr>
<td>dcust</td>
<td>SDO</td>
<td>Queries the Customer table to retrieve all customer records for the selected query. When this section is launched, it accepts the handle for the instance of dorder in the Order Tracking System main window.</td>
</tr>
</tbody>
</table>

**Note:** This module uses a handle passed in from the main window. This module does not run properly unless launched from the main window.
Order section

The Order section of the Order Tracking System contains SmartObjects for querying, viewing, adding, and changing order information.

The Order section has the following buttons:

- **Place New Order** — Places new orders for a customer
- **Find Order** — Opens a dialog box to search for order information
- **Maintain Order** — Opens a dialog box to view, update, and delete orders

**Placing new orders**

The **Place New Order** button in the Order section opens the **Order Entry / Maintenance** window to create new orders.

The **Order Entry / Maintenance** window is a nonmodal child window of the main window. As a result, you can move between the **Order Entry / Maintenance** window and the main window by clicking on either one.

The **Order / Entry Maintenance** window has an SDV for viewing order information and a SmartFolder with two tabs. A SmartFolder is a SmartObject that organizes the display of information into logical groupings, usually presenting the most frequently needed information on the first tab. In this instance, you can view different levels of detail about the customer’s orders and the individual lines of an order by choosing different tabs.
To view order information and place a new order:

1. Click the **Place New Order** button. The **Order Entry / Maintenance** window appears:

   ![Order Entry / Maintenance Window](image1.png)

   The **Order Header** tab contains a SDV that displays information about the order currently shown in the Order section of the Order Tracking System main window. The **Order Header** tab also contains a Navigation panel and a record Update panel.

2. Choose the **Order Lines** tab to view information about the order’s line items:
This tab contains an SDV that displays information about the individual order lines in the current order.

3. Click the **browse** button next to the **Item** field. The **Item Selection** window appears:

The **Item Selection** window is a SmartSelect Browser window with a dynamic SDB for browsing item information.

4. Close the windows and return to the main window.

Table 2–13 describes the major objects in the Order Entry/Maintenance section of the Order Tracking System.

Table 2–13: **Components of the Order Entry / Maintenance section**  

<table>
<thead>
<tr>
<th>Object or filename</th>
<th>Object type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>wmntord</td>
<td>SmartWindow</td>
<td>Serves as a container for the objects in the Order Entry/Maintenance section of this module.</td>
</tr>
</tbody>
</table>
| d_cust             | SDO         | Queries the Customer table to retrieve all customer records for the selected query. When this module is launched, it accepts the handle for the instance of **d_cust** in the Order Tracking System main window.  
**Note:** This module uses a handle passed in from the main window. This module does not run properly unless launched from the main window. |
| d_order            | SDO         | Queries the Order table to retrieve all order records for the selected query. When this module is launched, it accepts the handle for the instance of **d_order** in the Order Tracking System main window.  
**Note:** This module uses a handle passed in from the main window. This module does not run properly unless launched from the main window. |
| d_orderline        | SDO         | Queries the OrderLine table to retrieve all order line records for the selected query. When this module is launched, it accepts the handle for the instance of **d_orderline** in the Order Tracking System main window.  
**Note:** This module uses a handle passed in from the main window. This module does not run properly unless launched from the main window. |
### Table 2–13: Components of the Order Entry / Maintenance section

<table>
<thead>
<tr>
<th>Object or filename</th>
<th>Object type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>vcust</td>
<td>SDV</td>
<td>Displays the contents of several fields of the selected customer record.</td>
</tr>
<tr>
<td>folder</td>
<td>SmartFolder</td>
<td>Organizes the display of objects onto two logical pages, shown as the Order Header and Order Lines tabs.</td>
</tr>
<tr>
<td>vorder</td>
<td>SDV</td>
<td>Displays the contents of several fields from the Order and Salesrep tables for the selected order record.</td>
</tr>
<tr>
<td>dsalesrep</td>
<td>SDO</td>
<td>Queries the Salesrep table to retrieve all sales rep records for the selected query.</td>
</tr>
<tr>
<td>dynselect</td>
<td>SmartSelect</td>
<td>Allows the user to navigate through the records in dsalesrep.</td>
</tr>
<tr>
<td>pnavlb1</td>
<td>SmartPanel (navigation)</td>
<td>Allows the user to navigate through the current result set.</td>
</tr>
<tr>
<td>pupdsav</td>
<td>SmartPanel (update)</td>
<td>Allows the user to add, change, or delete records.</td>
</tr>
<tr>
<td>vordlne2</td>
<td>SDV</td>
<td>Displays the contents of several fields of the selected order line record.</td>
</tr>
<tr>
<td>ditem</td>
<td>SDO</td>
<td>Queries the Item table to retrieve item records for vordlne2.</td>
</tr>
<tr>
<td>dynselect</td>
<td>SmartSelect</td>
<td>Allows the user to navigate through the records in ditem.</td>
</tr>
<tr>
<td>btCustomerDiscount</td>
<td>Button</td>
<td>Runs getcustdiscount.p. Reads the returned value and then runs the calculatePrice internal procedure.</td>
</tr>
<tr>
<td>getcustdiscount.p</td>
<td>Procedure file</td>
<td>Returns the discount from the Customer table.</td>
</tr>
<tr>
<td>btBulkDiscount</td>
<td>Button</td>
<td>Runs getbulkdiscount.p. Reads the returned value and then runs the calculatePrice internal procedure.</td>
</tr>
<tr>
<td>getbulkdiscount.p</td>
<td>Procedure file</td>
<td>Returns the minimum quantities needed to qualify for a discount for each item.</td>
</tr>
<tr>
<td>pnavlb1-2</td>
<td>SmartPanel (navigation)</td>
<td>Allows the user to navigate a set of records in a SDO.</td>
</tr>
<tr>
<td>pupdsav-2</td>
<td>SmartPanel (update)</td>
<td>Allows the user to add, change, or delete records.</td>
</tr>
</tbody>
</table>
Finding orders

The **Find Order** button in the Order section opens the **Find Order** window to view order information.

The **Find Order** window is a nonmodal child window that is called from the main window. As a result, you can move between the Find Order window and the main window by clicking on either one.

The **Find Order** window contains a SmartFilter and a dynamic browser. Use the SmartFilter to define criteria to search through the selected customer’s orders. You can define a search query based on the date ordered, the order number, the order status, the sales representative, and the shipped date. Use the selection list to control which fields are displayed in the dynamic browser.

To view order information using a dynamic query:

1. Click the **Find Order** button. The **Find Order** window appears:

![Find Order Window](image)

   Initially, the window displays all available orders for the selected customer. You can use the **Order** filter section to define new search criteria to limit the orders that appear. You use the **Select Fields Displayed in Browser** selection list to define which fields you want to have displayed in the dynamic browser.

2. Type **01/01/98** into the **Shipped** field and choose the **Apply Filter** button.

   The browser now displays order information based on the shipped date you entered.

   **Note:** The fields in the browser remain the same as those in the initial search.
3. From the Select Fields Displayed in Browser list, choose the following fields for the browser:

- CustNum
- OrderDate
- Ordernum
- OrderStatus
- SalesRep
- ShipDate

4. Click the Re-display Browser button to display the new fields in the dynamic browser.

5. Close the window and return to the main window.

Table 2–14 describes the major objects in the Find Order section of the Order Tracking System.

**Table 2–14: Components of the Find Order section**

<table>
<thead>
<tr>
<th>Object or filename</th>
<th>Object type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>wfdord2.w</td>
<td>SmartWindow</td>
<td>Serves as a container for the other objects in Find Order section of this module.</td>
</tr>
<tr>
<td>dorder</td>
<td>SDO</td>
<td>Queries the Order table to retrieve all order records for the selected query. When this module is launched, it accepts the handle for the instance of dorder in the Order Tracking System main window. <strong>Note:</strong> This module uses a handle passed in from the main window. This module does not run properly unless launched from the main window.</td>
</tr>
<tr>
<td>getnewfields.p</td>
<td>Procedure file</td>
<td>Runs during the initializeObject internal procedure to populate the seldspflds selection list.</td>
</tr>
<tr>
<td>dynfilter</td>
<td>SmartFilter</td>
<td>Enables the user to dynamically filter the records shown in the browser.</td>
</tr>
<tr>
<td>seldspflds</td>
<td>Selection-list</td>
<td>Lists the fields in the Order table that can be displayed in the browser.</td>
</tr>
<tr>
<td>BUTTON-2</td>
<td>Button</td>
<td>Creates a dynamic browser instance from the master file, adm2\dynbrowser.w, using information from the filter and selection list.</td>
</tr>
<tr>
<td>dynbrowser</td>
<td>SDB</td>
<td>Enables the user to browse a result set created dynamically from dorder and information from the filter and selection list.</td>
</tr>
</tbody>
</table>
Changing orders

The Maintain Order button in the Order section opens the Order / Entry Maintenance window. This is the same module that you examined in the “Placing new orders” section on page 2–33. There are two entry points for this module to allow for differences in the way orders are created and updated.

If your business rules include differences in how orders are entered and how they are maintained, you would code custom triggers on these buttons to enforce those rules. For example, if only field agents could enter orders, you would code the Order Entry button to be disabled for anyone who was not a field agent.

Order Line section

The Order Line section of the Order Tracking System main window contains a SmartFolder with two tabs. One tab has an SDB for viewing Order Lines. The other tab has an SDV for viewing details about the Order Line. The display of the tabs in the SDV is controlled using logical pages. See the “Building the Order Line section” section on page 3–45 for more information about logical pages.

To view the tabs in the SmartFolder:

1. Choose the Browser tab and select a particular line of the order:

2. Choose the Detail tab to view the details of just that line of the order:
Menu items

Less commonly used modules of the Order Tracking System can be launched through menu options. The Order Tracking System has menu options for maintaining the product catalog, examining warehouse records, and setting the language in which the application displays.

To view the modules available from the menu:

1. Choose View → Item Maintenance. The Item Maintenance window appears:

   The Item Maintenance window is a SmartWindow with an SDB, an SDV, a Navigation SmartPanel, and an Update SmartPanel.

   **Note:** Table 2–7 describes the important elements of this window.

2. After examining this window, close it and return to the main window.
3. Choose View→Inventory Transfers. The Inventory Transfer dialog box appears:

The Inventory Transfer dialog box is a SmartDialog that is a child of the main window. The Inventory Transfer dialog box displays inventory records from the database. The fill-in fields enable you to specify search criteria to narrow the records shown in the SDB.

Table 2–15 describes the major objects in the Inventory Transfer section of the Order Tracking System.

Table 2–15: Components of the Inventory Transfer section

<table>
<thead>
<tr>
<th>Object or filename</th>
<th>Object type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>winvtrans.w</td>
<td>SmartWindow</td>
<td>Serves as a container for the objects in the Inventory Transfer section of this module</td>
</tr>
<tr>
<td>dinvtrans</td>
<td>SDO</td>
<td>Queries the InventoryTrans table for inventory information</td>
</tr>
<tr>
<td>binvtrans</td>
<td>SDB</td>
<td>Enables the user to select an inventory record</td>
</tr>
<tr>
<td>btn-search</td>
<td>Button</td>
<td>Uses any values in the fill-in fields to replace the WHERE clause in dinvtrans</td>
</tr>
</tbody>
</table>

4. Close the dialog and return to the main window.
5. Choose System Defaults → Set Application Language. The Set Application Language window appears:

![Set Application Language Window](image)

OpenEdge applications can be translated into many languages using the Translation Manager tool. This module enables each user to choose among the available translations for the application.

Table 2–16 describes the major objects in the Set Application Language section of the Order Tracking System.

<table>
<thead>
<tr>
<th>Object or filename</th>
<th>Object type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>wlang.w</td>
<td>SmartWindow</td>
<td>Serves as a container for the objects in the Set Application Language section</td>
</tr>
<tr>
<td>SELECT-1</td>
<td>Selection-list</td>
<td>Lists the available languages in which the module can display—selecting a new language sets it as the current language, if you have it installed on your machine</td>
</tr>
</tbody>
</table>

6. Close all the open SportsPro application windows.

Now that you have worked with the sample application, you have a sense of how a user would interact with an application and the applications that can be built using the AppBuilder and the SmartObjects. For a better understanding of how to use the AppBuilder to assemble the sample application, see Chapter 3, “Assembling the Order Tracking System.”

For an overview of SmartObject technology and information on fabricating your own SmartObjects, see Chapter 4, “Overview of SmartObject Technology.”
Assembling the Order Tracking System

This chapter is a tutorial that leads you through assembling the main window of the sample application, the SportsPro Order Tracking System. You will assemble it from the sample SmartObjects supplied with OpenEdge.

**Note:** The sample files are part of the Documentation and Samples CD that you received with your product. For more information, see the “Example procedures” section on page Preface–4.

To assemble the application, you start by creating a new SmartWindow to serve as a container for the other objects. Then you add the SmartObjects to the SmartWindow. As you do add more SmartObjects, you create SmartLinks that allow the various SmartObjects to exchange information and instructions with each other in a standardized way.

This chapter has the following sections:

- Progress Advisor
- Tutorial structure
- Reconnecting the Sports2000 database
- Assembling the main window
- Building the Customer section
- Building the Order section
- Building the Order Line section
Progress Advisor

While assembling the SportsPro Order Tracking System, you receive helpful information and suggestions on creating SmartLinks from the Progress Advisor. SmartLinks provide direct communication pathways between SmartObjects. The Progress Advisor is a dialog box that appears whenever you add a new SmartObject that could potentially serve as a source or target for a SmartLink.

The usefulness of the Progress Advisor’s suggestions depends on the way you plan your application’s development. To make the best use of the Advisor, plan your application around a required workflow and assemble the SmartObjects and other components in a way that reflects the workflow as closely as possible. For more information about SmartLinks and planning an application, see Chapter 4, “Overview of SmartObject Technology.”
The tutorial is divided into sections that outline a related group of tasks. Each section begins by showing you a screen shot of what your application should look like when you complete that part of the tutorial. The section then gives a list of tasks that you should be familiar with in order to complete the section.

If you already know how to complete these tasks, you should try the exercise on your own. Below the task list is a condensed set of steps to follow. These steps provide you with basic information you need, such as the dimensions of the objects you should build. If you want to review a particular procedure in detail, the cross-references under each step enable you to find that procedure in the detailed tutorial that follows.

The final part of each section is a detailed tutorial. This tutorial leads you through the exercise for each section in detail appropriate to a developer who is new to the OpenEdge Studio but has some familiarity with graphical user interface (GUI) design environments.
Reconnecting the Sports2000 database

Each time you restart your OpenEdge session, you need to reconnect to the Sports2000 database. There is a simpler method than using the Data Dictionary available in the AppBuilder.

To reconnect the Sports2000 database:

1. Choose **Tools** → **Database Connections** from the AppBuilder menu. The **Database Connections** dialog box appears:

2. Click **Connect**. The **Connect Database** dialog box appears:

3. Use the **Browse** button to select your copy of the **Sports2000 database** from your working directory.

4. Click **OK** and then close the **Database Connections** dialog box.
Assembling the main window

The first stage of creating the SportsPro Order Tracking System module is building the main window and dividing it into sections. When you are finished with this stage, your window should look something like the following:

![Main Window Diagram](image)

**Task list and short procedure**

To reach this point, you must know how to perform the following tasks:

1. Access the **New** dialog box.
2. Access an object’s **Property Sheet**.
3. Use your mouse to move and resize an object.
4. Add an image to a frame.
5. Create and resize a rectangle.
6. Create text labels.
7. Access the **List Objects** dialog box.
8. Save your file.

**Note:** During application assembly, the main window is sometimes referred to as the workspace or design window.
The following short procedure outlines how to complete these tasks.

To assemble the main window:

1. Create a new SmartWindow. See the “Detailed main window procedure” section on page 3–6 to review this procedure.

2. Set it to be 147.00 units wide and 25.50 units high. See Step 3 in the “Detailed main window procedure” section on page 3–6 to review this procedure.

3. Resize the default frame. See Step 5 in the “Detailed main window procedure” section on page 3–6 to review this procedure.

4. Add the application banner. See the “Adding a banner image” section on page 3–9 to review this procedure.

5. Divide the window into three sections using rectangles with the following properties:

<table>
<thead>
<tr>
<th>Column</th>
<th>Row</th>
<th>Width</th>
<th>Height</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.00</td>
<td>2.1</td>
<td>147</td>
<td>5.48</td>
</tr>
<tr>
<td>1.00</td>
<td>7.48</td>
<td>147</td>
<td>8.33</td>
</tr>
<tr>
<td>1.00</td>
<td>15.81</td>
<td>147</td>
<td>10.67</td>
</tr>
</tbody>
</table>

See the “Outlining the Customer, Order, and Order Line sections” section on page 3–12 to review this procedure.

6. Add dark blue text labels naming the three sections: **Customer, Order, Order Line**. See the “Adding text labels to the sections” section on page 3–13 to review this procedure.

7. Set the title of your window to **SportsPro Order Tracking System**. See the “Naming and saving the new SmartWindow” section on page 3–15 to review this procedure.

8. Save your window in a file named *wordtrk.w*. See Step 5 in the “Naming and saving the new SmartWindow” section on page 3–15 to review this procedure.

**Detailed main window procedure**

The sections that follow guide you through assembling your main window in more detail.

**Creating a SmartWindow**

The first step in assembling the main window is to create a new SmartWindow. The SmartWindow serves as a container for other SmartObjects.
To create a new SmartWindow:

1. Open the New dialog box and select SmartWindow:

A new, untitled SmartWindow and the SmartWindow Cue Card appear:

Each time you create a new SmartObject, the AppBuilder displays a Cue Card for the type of SmartObject you are creating. The Cue Card provides useful information including an explanation of how the SmartObject works and key steps to follow while creating that SmartObject. You do not need to use the Cue Cards for this tutorial.

Remember, when you use the AppBuilder to develop an application, the Cue Cards are available to assist you. If you close the Cue Cards, you can always open them again using the Help → Cue Card option from the AppBuilder menu bar.
2. Click the **Object Properties** icon in the AppBuilder toolbar. The property sheet for the new *widget* appears. Widget is a general term for a visual element of the GUI. The basic property sheet allows you to specify the column, row, height, width in character units, and other object-specific attributes as shown in the following figure:

![Property Sheet](image)

The **Advanced** button opens the **Advanced Property** dialog box with additional properties.

3. Type **147.00** for the **Width** and **25.50** for the **Height** in the **Geometry** section. These dimensions resize the window so that it is large enough for the objects you will add.

4. Click **OK** in the property sheet. The main window resizes to its new dimensions:
Notice that the default frame retained the original dimensions of the SmartWindow. Because SmartContainers can contain more than one frame, their default frames do not automatically resize when you expand the SmartContainer. The frames in design windows for other types of SmartObjects do resize automatically with the window.

5. Use the frame’s **handles** to fit the frame to the SmartWindow:

You are now ready to define the sections of the main window. When you ran the sample application in Chapter 2, “Working with the Sample Application,” you saw that the Order Tracking System main window is divided into these sections:

- **Banner**
- **Customer**
- **Order**
- **Order Line**

**Adding a banner image**

In this section, you add a graphic banner to the main window. After adding the banner, you define the Customer, Order, and Order Line sections.
To add a banner to the main window:

1. Click the Image button in the Object Palette and place one on your window. A placeholder image appears in the main window:

2. Double-click the placeholder image. The IMAGE–1 property sheet appears:

3. Click the Image button. The Choose Image Files dialog box appears.

4. Browse to and select `<wrk>\images\s2_banr.jpg`, where `<wrk>` is your working directory.
5. Click Open. The File list displays all the images in the directory. The selected image previews to the right of the file list:

Note: The image appears in the Preview section only if you have selected the Preview option.

6. Click OK, and close the property sheet to return to your workspace.

7. Resize the banner and position it, as shown in the following figure:
Outlining the Customer, Order, and Order Line sections

After creating the main window and adding the banner, your next task is to divide the window into Customer, Order, and Order Line sections.

To outline the window's sections:

1. Double-click the Rectangle icon in the Object Palette. A lock appears on the icon indicating that you can add multiple rectangles without reselecting the icon for each.

   **Note:** To identify an object in the palette, select the Object Palette and pass the cursor over an object. As the pointer passes over an object, a ToolTip identifies that object. For more information about the Object Palette, see Chapter 1, “Overview of the OpenEdge Studio.”

2. Draw rectangles to roughly outline the Customer, Order, and Order Line sections.

3. Double-click each rectangle in turn to open its property sheet:

4. Set the following values for each rectangle’s Geometry attributes:

<table>
<thead>
<tr>
<th>Rectangle</th>
<th>Column</th>
<th>Row</th>
<th>Width</th>
<th>Height</th>
</tr>
</thead>
<tbody>
<tr>
<td>RECT-1</td>
<td>1.00</td>
<td>2.10</td>
<td>147</td>
<td>5.48</td>
</tr>
<tr>
<td>RECT-2</td>
<td>1.00</td>
<td>7.48</td>
<td>147</td>
<td>8.33</td>
</tr>
<tr>
<td>RECT-3</td>
<td>1.00</td>
<td>15.81</td>
<td>147</td>
<td>10.67</td>
</tr>
</tbody>
</table>
Assembling the main window

The AppBuilder resizes the rectangles to reflect the new settings. Your main window should now look like this:

Adding text labels to the sections

Now that you have divided your main window into sections for Customer, Order, and Order Line information, you need to add text labels to identify the sections.

To add text labels to the sections:

1. Double-click the Text icon in the Object Palette.

2. Place a text label in each section, as shown:
3. Open the property sheet for the text label in the first section:

4. Type Customer in the Text field:

5. Click the Color icon to open the Choose Color dialog box:

6. Select dark blue, setting 1, for the Foreground Colors.

   Note: The colors labeled with a question mark indicate the default colors.

7. Click OK and return to the main window.

8. Resize the text label so that all the text is visible.
9. Repeat Step 3 through Step 8 for the **Order** and **Order Line** text labels. When you are finished, the window looks like the following:

![Main window with repeated order and order line labels](image)

**Naming and saving the new SmartWindow**

Now that you have your main window’s basic design, you should name the SmartWindow and save your work before adding SmartObjects to it.

To name the SmartWindow and save your work:

1. Click the **List Objects** icon on the **AppBuilder toolbar**. The **List Objects** dialog box appears:

![List Objects dialog box](image)
2. Double-click the **wWin** object from the list to open the property sheet:

![Property Sheet - wWin](image.png)

3. Type **SportsPro Order Tracking System** for the Title and click **OK**.

4. Close the property sheet. The title appears in the title bar:

![Property Sheet - wWin](image.png)

5. Save your file as **wordtrk.w** in your working directory.

You are now ready to add objects to your window.
Building the Customer section

The Customer section of the SportsPro Order Tracking System contains a SmartDataObject (SDO) for querying customer information, a SmartDataViewer (SDV) for viewing customer information, and buttons to launch the Find Customer and Customer Maintenance modules. When you are finished with this section of the tutorial, your window should look something like the following:

Task list and short procedure

To reach this point, you must know how to perform the following tasks:

1. Add a prebuilt SDO
2. Add a prebuilt SDV
3. Link an SDV to an SDO
4. Create and resize a button
5. Enhance a button with a ToolTip, a Help ID, and an image
6. Link a Help file to a window
7. Code a launch trigger for a button using the Section Editor

The following short procedure outlines how to complete these tasks:

To build the Customer section:

1. Add a prebuilt SDO, `dcust.w`, to the Customer section. See the “Adding an SDO” section on page 3–18 to review this procedure.

2. Add a prebuilt SDV, `vcust.w`, to the Customer section. See the “Adding an SDV” section on page 3–20 to review this procedure.
3. Create a Data SmartLink from the SDO to the SDV. See Step 3 in the “Adding an SDV” section on page 3–20 to review this procedure.

4. Create an Update SmartLink from the SDV to the SDO. See Step 4 in the “Adding an SDV” section on page 3–20 to review this procedure.

5. Create two buttons with the following properties:

<table>
<thead>
<tr>
<th>Object</th>
<th>Left</th>
<th>Height</th>
<th>Width</th>
<th>ToolTip</th>
<th>Help ID</th>
<th>Image</th>
</tr>
</thead>
<tbody>
<tr>
<td>BUTTON-1</td>
<td>14</td>
<td>2.38</td>
<td>10</td>
<td>Find Customer</td>
<td>1</td>
<td>Findcust.jpg</td>
</tr>
<tr>
<td>BUTTON-2</td>
<td>14</td>
<td>2.38</td>
<td>10</td>
<td>Maintain Customer</td>
<td>2</td>
<td>mntcust.jpg</td>
</tr>
</tbody>
</table>

**Note:** The image files are stored in the images subdirectory of your working directory.

See the “Creating and positioning the buttons” section on page 3–23 and the “Adding ToolTips, help IDs, and images to the buttons” section on page 3–25 to review this procedure.

6. Link a help file to your window. See the “Adding online help to the window” section on page 3–27 to review this procedure.

7. Write `CHOOSE` event triggers for the buttons in the Section Editor. See the “Coding event triggers for the buttons” section on page 3–28 to review this procedure.

**Detailed Customer section procedure**

The following sections lead you through assembling the Customer section in more detail.

**Adding an SDO**

The first SmartObject added in most applications is the SDO that acts as a data source for the rest of the application’s SmartObjects.
To add an SDO:

1. Click the SDO icon in the Object Palette. The Choose SmartDataObject dialog box appears:

2. Select dcust.w and choose OK.

3. Move the pointer over the Customer section. Note that your pointer resembles the SDO icon.

4. Click once in the Customer section. You do not need to establish an exact position for an SDO because it is invisible when the application is running. The new SDO appears in the Customer section:
Adding an SDV

Next, you need to add an SDV. The SDV displays the customer information retrieved by the SDO.

To add an SDV:

1. Click the SDV icon in the Object Palette. The Choose SmartDataViewer dialog box appears:

```
Choose SmartDataViewer
```

2. Select vcust.w and click OK.

3. Click once in the Customer section. Note that the pointer now resembles the SDV icon. The Progress Advisor notifies you that the SDV instance, h_vcust, should be a data target for another SmartObject, and offers to create a Data SmartLink from the SDO instance, h_dcust:

```
Progress Advisor
```

**Note:** The Progress Advisor refers to SmartObjects with an h_ prefix to indicate a handle. A handle refers to an instance of a SmartObject in memory. For example, vcust is referred to by its handle, h_vcust.
4. Select the radio button to create the SmartLink and choose OK. The Progress Advisor notifies you that the SDV instance, h_vcust, should be an Update Source for some other SmartObject, and offers to create an Update SmartLink to the SDO instance, h_dcust:

![Progress Advisor](image1)

5. Select the radio button to create the SmartLink and click OK. The SDV, vcust, appears in the Customer section.

6. Position the SDV within the Customer section, aligning it on the right edge of the design window, as shown in the following figure:

![SDV within Customer section](image2)

7. Double-click the SDV to open its property sheet:

![Property Sheet - h_vcust](image3)
8. Click the **SmartLinks** button to view the **SmartLinks** dialog box. It shows a Data SmartLink from h_dcust to h_vcust and an Update SmartLink from h_vcust to h_dcust:

You can check the SmartLinks between your objects with this dialog. You can add, modify, and remove SmartLinks as needed. This allows you to alter how information flows between the SmartObjects in your application.

9. Close the open dialog box boxes and save your work.

Your next task is to add two buttons to the Customer section, complete with ToolTips, images, and help text.
Creating and positioning the buttons

The first step in adding buttons to your application is to create and position them.

To create and position the buttons:

1. Double-click the **Button** icon in the **Object Palette**.

2. Click in the **Customer** section in two places to add two buttons, as shown. You do not need to position the buttons exactly. You will use the **Properties Window** to specify precise settings later. The main window should look something like the following:

3. Click the **Pointer** icon and, holding down the **CTRL** key, select both buttons.

4. Choose **Window → Properties Window** from the **AppBuilder** menu. The **Properties Window**, which enables quickly setting common attributes in many objects at once, appears:
5. Double-click the **Geometry** row to expand its options:

![Properties Window]

**Note:** In this **Properties Window**, you can only change one value at a time. Select a row, type a value in the **fill-in** field next to the **check-mark** button, and then choose the **check-mark** to commit the change.

6. Set the following attributes for the buttons:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Left</td>
<td>14</td>
</tr>
<tr>
<td>Height</td>
<td>2.38</td>
</tr>
<tr>
<td>Width</td>
<td>10</td>
</tr>
</tbody>
</table>

When you finish, the window should look like the following:

![Properties Window (updated)]
7. Close the **Properties Window** and manually adjust the buttons’ vertical positions to resemble the following figure:

![Image of buttons with properties](image)

### Adding ToolTips, help IDs, and images to the buttons

After creating and positioning the buttons, you are ready to add a ToolTip, a Help ID, and an image to each button.

**To add ToolTips, Help IDs, and images to the buttons:**

1. Open the property sheet for **BUTTON-1** and set the following properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tooltip</td>
<td>Find Customer</td>
</tr>
<tr>
<td>Help ID</td>
<td>1</td>
</tr>
</tbody>
</table>

2. Click the **Up** button in the **Images** section. The **Choose Image** dialog box appears.
3. Select `Findcust.jpg` from `<wrk>\images`, where `<wrk>` is your OpenEdge working directory. The button’s property sheet now looks like this:

![Property Sheet - BUTTON-1](image)

4. Repeat the procedure for `BUTTON-2`, setting the following properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tooltip</td>
<td>Maintain Customer</td>
</tr>
<tr>
<td>Help ID</td>
<td>2</td>
</tr>
<tr>
<td>Up Image</td>
<td>mntcust.jpg</td>
</tr>
</tbody>
</table>

When you finish, your window should appear as follows:
Adding online help to the window

Now that you have added Help IDs, you can link a context-sensitive help file to your application. Help files are added to the window. All of the simple objects on that window reference the same context-sensitive help.

To add a context-sensitive help file:

1. Click the List Object icon on the AppBuilder toolbar. The List Objects dialog box appears:

   ![List Objects dialog box](image)

2. Select wWin from the Name list to select the window object.

3. Click the Object Properties icon in the AppBuilder toolbar to open the window’s property sheet:

   ![Property Sheet - wWin](image)
4. Select the Context Help toggle-box to activate context sensitive help.

5. Type sports2000.hlp in the Help File field and choose OK.

6. Close the List Objects dialog box and save your window.

Coding event triggers for the buttons

Your next step is to add code to the buttons. When the user clicks on either button, a CHOOSE event occurs. Using the Section Editor, you can quickly add code to launch the appropriate module when each CHOOSE event occurs.

To add the trigger code to the buttons:

1. Select BUTTON-1 and click the Edit Code icon in the AppBuilder toolbar. The Section Editor appears:

   ![Section Editor](image)

2. Between the DO and END statements, add the following code:

   ```
   RUN wfndcust.w (h_dcust).
   ```

Note: This statement passes the handle for the dcust SDO to the module you are starting. As a result, the new module opens on the same customer record. Because the new module does not have an internal data source, it cannot run on its own.
3. Select **BUTTON-2** in the combo box of objects:

![Section Editor screenshot](image)

4. Between the **DO** and **END** statements, add the following code:

```
RUN wmntcust.w (h_dcust).
```

5. Choose **File → Close Window** to save the new code and return to the workspace.

**Note:** If you do not want to save code that you entered in the **Section Editor**, you must use the **Edit → Undo** menu options.

6. Save your work.

Now that you have built the Customer section of the SmartWindow, you can add SmartObjects to the Order section.
Building the Order section

The Order section contains SmartObjects for querying and viewing order information: SDOs, an SDV, Object Component Extension (OCX) controls (also known as ActiveX controls), and buttons for launching associated modules. When you are finished with this section of the tutorial, your window should look something like the following:

![Order section window](image)

**Task list and short procedure**

To reach this point, you must know how to perform the following tasks:

1. Add a prebuilt SDO
2. Link SDOs together and map their foreign fields
3. Add a prebuilt SDV
4. Link an SDV to an SDO
5. Create and resize a button
6. Enhance a button with a ToolTip, a Help ID, and an image
7. Code a launch trigger for a button using the Section Editor
8. Create and resize an OCX control
9. Code navigation triggers for an OCX control using the Section Editor
The following short procedure outlines how to complete these tasks.

To build the Order section:

1. Add a prebuilt SDO, dorder.w, to the Order section. See the “Adding an SDO for orders” section on page 3–32 to review this procedure.

2. Create a Data SmartLink from dcust.w to dorder.w. See Step 3 in the “Adding an SDO for orders” section on page 3–32 to review this procedure.

3. Map the Customer.CustNum field to the Order.CustNum field. See Step 4 in the “Adding an SDO for orders” section on page 3–32 to review this procedure.

4. Add a prebuilt SDO, dordline.w, to the Order section. See the “Adding an SDO for order lines” section on page 3–34 to review this procedure.

5. Create a Data SmartLink from dorder.w to dordline.w. See Step 3 in the “Adding an SDO for order lines” section on page 3–34 to review this procedure.

6. Map the Order.OrderNum field to the OrderLine.OrderNum field. See Step 5 in the “Adding an SDO for order lines” section on page 3–34 to review this procedure.

7. Add a prebuilt SDV, vorder.w, to the Order Section. See the “Adding an SDV” section on page 3–35 to review this procedure.

8. Create a Data SmartLink from dorder.w to vorder.w. See Step 3 in the “Adding an SDV” section on page 3–35 to review this procedure.

9. Create an Update SmartLink from vorder.w to dorder.w. See Step 4 in the “Adding an SDV” section on page 3–35 to review this procedure.

10. Add three buttons with the following properties:

<table>
<thead>
<tr>
<th>Object</th>
<th>Left</th>
<th>Height</th>
<th>Width</th>
<th>ToolTip</th>
<th>Help ID</th>
<th>Image</th>
</tr>
</thead>
<tbody>
<tr>
<td>BUTTON-3</td>
<td>14</td>
<td>2.38</td>
<td>10</td>
<td>Place New Order</td>
<td>3</td>
<td>crtord.jpg</td>
</tr>
<tr>
<td>BUTTON-4</td>
<td>14</td>
<td>2.38</td>
<td>10</td>
<td>Find Order</td>
<td>4</td>
<td>findord.jpg</td>
</tr>
<tr>
<td>BUTTON-5</td>
<td>14</td>
<td>2.38</td>
<td>10</td>
<td>Maintain Order</td>
<td>5</td>
<td>mainord.jpg</td>
</tr>
</tbody>
</table>

See the “Creating and positioning the buttons” section on page 3–23 and the “Adding ToolTips, help IDs, and images to the buttons” section on page 3–25 to review this procedure.

11. Write CHOOSE event triggers for the buttons in the Section Editor. See the “Coding event triggers for the Order buttons” section on page 3–37 to review this procedure.

12. Add and position two control frames, based on the Crescent Spin Control OCX. See the “Adding OCX controls” section on page 3–37 to review this procedure.
13. Assign the **OCX control frames** the following properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Height</td>
<td>1.52</td>
</tr>
<tr>
<td>Width</td>
<td>9.0</td>
</tr>
<tr>
<td>Style</td>
<td>3D Horizontal</td>
</tr>
<tr>
<td>ArrowColor</td>
<td>dark blue</td>
</tr>
</tbody>
</table>

See the “Changing the appearance of the OCX controls” section on page 3–39 to review this procedure.

14. Create the following **Text** labels:

<table>
<thead>
<tr>
<th>Object</th>
<th>Label</th>
<th>Color</th>
</tr>
</thead>
<tbody>
<tr>
<td>CtrlFrame</td>
<td>Prev/Next</td>
<td>1</td>
</tr>
<tr>
<td>CtrlFrame-2</td>
<td>First/Last</td>
<td>1</td>
</tr>
</tbody>
</table>

See Step 7 in the “Changing the appearance of the OCX controls” section on page 3–39 to review this procedure.

15. Place a rectangle around each OCX control and its text label. See Step 8 in the “Changing the appearance of the OCX controls” section on page 3–39 to review this procedure.

16. Add **OCX.SpinDown** and **OCX.SpinUp** triggers for the control frames. See the “Coding the Prev/Next spin box to fetch the previous record” section on page 3–41 and the “Coding the First/Last spin box to fetch the first and last records” section on page 3–43 to review this procedure.

17. Run and check your application. See the “Viewing the window” section on page 3–44 to review this procedure.

**Detailed Order section procedure**

The sections that follow describe building the Order section in detail.

**Adding an SDO for orders**

As in the Customer section, the SDOs are the first objects to add to the Order section. The Order section needs SDOs that query the database for orders and the order lines that make up that order.
To add an Order SDO:

1. Click the SDO icon in the Object Palette to open the Choose SmartDataObject dialog box.

2. Select dorder.w and click OK.

3. Click once in the Order section. You do not need to position a SDO precisely, since it is invisible when an application is running. The Progress Advisor notifies you that the SDO instance, h_dorder, can be a Data Target for some other SmartObject, and offers to add a Data SmartLink from the SDO instance, h_dcust.

4. Select the radio button to create the SmartLink and click OK. The Progress Advisor notifies you that h_dorder and h_dcust are both Query Objects, and asks whether you want to specify foreign fields:

5. Select the radio button to choose foreign fields and click OK. The Multi-Field Mapping dialog box appears.

6. Select CustNum in the Source and Order.CustNum in the Target list:

7. Click Map. The field pair is displayed in the Mapped Fields field.

8. Click OK. The new SDO, dorder.w, appears in the main window.
Adding an SDO for order lines

You are now going to add another SDO, dord1ne, to the Order section.

To add an Order Lines SDO:

1. Click the SDO icon in the Object Palette to open the Choose SmartDataObject dialog box.
2. Select dord1ne.w and click OK.
3. Place the new SDO under the other SDO. This is only for convenience, their positions do not affect how they operate. The Progress Advisor notifies you that the SDO instance, h_dord1ne, can be a Data Target for some other SmartObject, and offers to add a Data SmartLink from either of the other SDO instances:

   ![Progress Advisor](image)

4. Select the radio button to create the SmartLink to h_dorder and click OK. The Progress Advisor notifies you that h_dord1ne and h_dorder are both Query Objects and asks if you want to choose foreign fields.
5. Select the radio button to select foreign fields and then click OK. The Multi-Field Mapping dialog box appears.
7. Click Map. The field pair is displayed in the Mapped Fields field.
8. Click OK. The new SDO, dordline, appears in the design window:

![Image of design window]

9. Save your work.

**Adding an SDV**

The next object to add is an SDV that displays the order information.

To add an SDV for displaying order information:

1. Click the SDV icon in the Object Palette to open the Choose SmartDataViewer dialog box.

2. Select vorder.w and click OK.

3. Click within the Order section to add the SDV. The Progress Advisor notifies you that SDV instance, h_vorder, should be a Data Target for some other SmartObject, and offers to create a Data SmartLink from the SDO instance, h_dorder.

4. Select the radio button to create the SmartLink and choose OK. The Progress Advisor notifies you that the SDV instance, h_vorder, should be an Update source for some other SmartObject, and offers to add an Update SmartLink to the SDO instance, h_dorder.

5. Select the radio button to create the SmartLink and click OK. The SDV appears in the main window.
6. Position the SDV near the top of the Order section and aligned on the right edge. Your window should look something like the following:

![Order section window](image)

**Adding buttons to the Order section**

You need to add buttons to the Order section, as you did to the Customer section. Follow the procedures in the “Creating and positioning the buttons” section on page 3–23 and the “Adding ToolTips, help IDs, and images to the buttons” section on page 3–25 to add and customize the three buttons to the Order section.

To customize the buttons, use the values in the following table:

<table>
<thead>
<tr>
<th>Object</th>
<th>Left</th>
<th>Height</th>
<th>Width</th>
<th>ToolTip</th>
<th>Help ID</th>
<th>Image</th>
</tr>
</thead>
<tbody>
<tr>
<td>BUTTON-3</td>
<td>14</td>
<td>2.38</td>
<td>10</td>
<td>Place New Order</td>
<td>3</td>
<td>crtord.jpg</td>
</tr>
<tr>
<td>BUTTON-4</td>
<td>14</td>
<td>2.38</td>
<td>10</td>
<td>Find Order</td>
<td>4</td>
<td>findord.jpg</td>
</tr>
<tr>
<td>BUTTON-5</td>
<td>14</td>
<td>2.38</td>
<td>10</td>
<td>Maintain Order</td>
<td>5</td>
<td>mainord.jpg</td>
</tr>
</tbody>
</table>

**Note:** You do not need to add a Help file for the Order section buttons. The Help file you added while building the Customer section controls the Help for the entire window. It applies to all sections of the window.
When you finish, your window should look like the following:

![Image of order section window]

**Coding event triggers for the Order buttons**

You need to add CHOOSE event triggers to the buttons, as you did in the Customer section. Follow the procedures in the “Coding event triggers for the buttons” section on page 3–28.

Using the Section Editor, add the following code to the CHOOSE events of the buttons in this section:

<table>
<thead>
<tr>
<th>Button</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>BUTTON-3</td>
<td>RUN wmntord.w (h_dcust, h_dorder, h_dord1ne).</td>
</tr>
<tr>
<td>BUTTON-4</td>
<td>RUN wfnord2.w(h_dorder).</td>
</tr>
<tr>
<td>BUTTON-5</td>
<td>RUN wmntord.w (h_dcust, h_dorder, h_dord1ne).</td>
</tr>
</tbody>
</table>

The remaining objects to add to the Order section are two Object Component Extension (OCX) controls, also known as ActiveX controls.

**Adding OCX controls**

In this section, you add Object Component Extension (OCX) controls to your window. An OCX (also called an ActiveX control) is a predefined object widget that you include in a control-frame to add prepackaged functionality to your applications.

ActiveX objects are Common Object Model (COM) objects that provide a form of interprocess communication with other Windows applications along the Microsoft ActiveX specifications. For more information about ActiveX Automation, see *OpenEdge Development: Programming Interfaces*. 
Assembling the Order Tracking System

The OpenEdge Studio ships with three licensed OCX controls:

- Crescent Spin Control, csspin32.ocx, a spin box
- CS ComboBox Control, cscomb32.ocx, a combo box
- Progress Timer Control, pstimer.ocx, an object that allows you to determine the length of time used by a process

With the AppBuilder, you can integrate third-party objects, like OCXs, into your applications in the same way you build SmartObjects. For example, in this section, you choose the OCX spin box from the Object Palette, position and resize it visually in the window, and write user-interface triggers for it using the Section Editor. For more detailed information about ActiveX (OCX controls) and the AppBuilder, see OpenEdge Development: AppBuilder.

Selecting an OCX object

In this section, you add two OCX spin box controls to the Order section. These OCX controls are used to navigate through the records in the SDO, dorder, which sends the order information to the SDV, vorder, for viewing.

To add OCX controls to the Order section:

1. Double-click the OCX icon in the Object Palette. The Choose Control dialog box appears:

   ![Choose Control dialog box](image)

   Note: If you installed other OCX controls, they should also appear in the Available Controls list.

2. Select the Crescent Spin Control in the Available Controls list. When you select an OCX control, the location and control name appear at the bottom of the Choose Control dialog box.

3. Click OK to return to your design window.
4. Add two spin boxes to the Order section. The spin boxes appear with their default dimensions. Your window should appear similar to the following:

![Spin Box Example]

**Changing the appearance of the OCX controls**

The OCX spin boxes are too large and oriented in the wrong direction. You need to resize the OCX spin boxes, change their orientation, and add text labels and borders.

To resize the OCX spin boxes and change their orientation:

1. Use the Window → Properties Window option in the AppBuilder menu to set the control frames’ **Heights** to 1.52 and their **Widths** to 9.0.

   See the “Creating and positioning the buttons” section on page 3–23 for information on using the Properties Window.

2. Double-click the left spin box to open the Property Editor dialog box:

   ![Property Editor Example]

3. Select the **Style** property and set it to **Style 3 - 3D Horizontal**.
4. Select the **ArrowColor** property and set it to **dark blue**.

5. Close the property sheet.

6. Repeat Step 2 through Step 5 for the **right spin box**. Your window should now look something like the following:

![Image of a window with a spin box]

7. Use the **Text** icon in the **Object Palette** to add the colored text labels, **Prev/Next**, below the **left spin box** and **First/Last** below the **right spin box**. See the “Adding text labels to the sections” section on page 3–13 to review this procedure.

8. Use the **Rectangle** icon in the **Object Palette** to add a rectangle around each spin box:

![Image of a window with a rectangle around a spin box]
Coding the Prev/Next spin box to fetch the previous record

You need to code the OCX controls in your window to navigate through the Order records. First, you will code the Prev/Next spin box to fetch the previous record.

To add trigger code to fetch the previous record:

1. Select the **Prev/Next spin box** and open the **Section Editor**.

2. Click the **New** button to open the **Choose Event** dialog box:

3. Select **OCX.SpinDown** from the **Common OCX Events** list. The **Section Editor** now displays the new code section:

4. Position the cursor between the **Header comments** and the **END PROCEDURE** statement.
5. Click the **Insert Call** button to open the **Insert Procedure Call** dialog box:

6. Select **h_dorder** from the **Procedure Object** drop-down list.

7. Select the **Procedures** radio button in the **Entries in Object** section, and select **fetchPrev**. The **Code to Insert** field shows a preview of the code for this procedure call:
8. Click OK. The new code appears in the Section Editor:

![Section Editor](image)

Coding the Prev/Next spin box to fetch the next record

Now, you need to code the Prev/Next spin box to fetch the next record.

To add trigger code to fetch the next record:

1. Click New in the Section Editor to open the Choose Event dialog box.
2. Select OCX.SpinUp in the Common OCX Events list.
3. Position the cursor between the Header comments and the END PROCEDURE statement.
4. Click the Insert Call button to open the Insert Procedure Call dialog box.
5. Select h_dorder from the Procedure Object drop-down list.
6. Select the Procedures radio button in the Entries in Object section.
7. Select fetchNext and click OK. The new trigger code appears in the Section Editor.
8. Choose File → Close Window to save the new code and close the Section Editor.

Coding the First/Last spin box to fetch the first and last records

The second spin box fetches the first or last record in the Order SDO. The only thing that differs from what you did for the Prev/Next spin box is the procedure called by the spin events. First, you add the code to fetch the first record.

To add trigger code to fetch the first record, repeat the procedure in the “Coding the Prev/Next spin box to fetch the previous record” section on page 3–41. Use the fetchFirst procedure, instead of the fetchPrev procedure.
Assembling the Order Tracking System

Now, you add the code to fetch the last record.

To add trigger code to fetch the last record, repeat the procedure in “Coding the Prev/Next spin box to fetch the next record” section on page 3–43. Use the `fetchLast` procedure, instead of the `fetchNext` procedure.

You can now save your window and have a look at how it runs so far.

**Viewing the window**

You have now completed assembling the Customer and Order sections of the application’s main window. If you want to view and test the partially assembled application, run the application to view the window at this point in the assembly. Before you run the application, save your work.

To view the partially assembled window:

1. Click the Run icon on the AppBuilder toolbar. Your window should appear as it would in a running application. The main window of the sample application should look like the following:

2. Hold your pointer over a button to view the ToolTip.

3. Click the context sensitive help button (?) in the upper-right corner of the window and then click a button to see the context sensitive help.

4. Select one of the buttons to launch another module.

5. Choose the Stop icon on the AppBuilder toolbar to return to the main window.

Now you can proceed to building the Order Line section.
Building the Order Line section

The Order Line section contains a SmartFolder with SmartObjects for viewing and browsing order line information. When you are finished with this section of the tutorial, your window should look like the following:

![Order Line section](image)

Task list and short procedure

To reach this point, you must know how to perform the following tasks:

1. Add a SmartFolder
2. Link the SmartFolder to the SmartWindow
3. Define two tabs on the SmartFolder
4. Set logical pages
5. Add a prebuilt SmartDataBrowser (SDB) to Page 1
6. Link an SDB to an SDO
7. Add a prebuilt SDV to Page 2
8. Link an SDV to an SDO
9. Add a Navigation SmartPanel to Page 2
10. Link the SmartPanel to the SDO
The following short procedure outlines how to complete these tasks.

To build the Order Line section:

1. Add a SmartFolder to the Order Line section. See the “Adding a SmartFolder” section on page 3–47 to review this procedure.

2. Create a Page SmartLink from the SmartFolder to the SmartWindow. See Step 2 in the “Adding a SmartFolder” section on page 3–47 to review this procedure.

3. Resize the SmartFolder. See Step 4 in the “Adding a SmartFolder” section on page 3–47 to review this procedure.

4. Create Browser and Detail tabs for the SmartFolder. See Step 5 in the “Adding a SmartFolder” section on page 3–47 to review this procedure.

5. Assign a logical page to the Browser tab and set it to display on startup. See the “Creating page 1 of the SmartFolder” section on page 3–50 to review this procedure.

6. Add the prebuilt SDB, bordline2.w, to the Browser tab. See the “Adding an SDB to page 1 of the SmartFolder” section on page 3–52 to review this procedure.

7. Create a Data SmartLink from the SDO, dordline.w, to the SDB. See Step 3 in the “Adding an SDB to page 1 of the SmartFolder” section on page 3–52 to review this procedure.

8. Create an Update SmartLink from the SDB to the SDO, dordline.w. See Step 4 in the “Adding an SDB to page 1 of the SmartFolder” section on page 3–52 to review this procedure.

9. Add the prebuilt SDV, vordline.w, to the Details tab. See the “Adding an SDV to page 2 of the SmartFolder” section on page 3–53 to review this procedure.

10. Create a Data SmartLink from the SDO, dordline.w, to the SDV. See Step 5 in the “Adding an SDB to page 1 of the SmartFolder” section on page 3–52 to review this procedure.

11. Add a Navigation SmartPanel to the Details tab. See the “Adding a Navigation SmartPanel” section on page 3–54 to review this procedure.

12. Create a Navigation SmartLink from the SmartPanel to the SDO, dordline.w. See Step 3 in the “Adding a Navigation SmartPanel” section on page 3–54 to review this procedure.

Detailed Order Line section procedure

The sections that follow guide you through assembling the Order Line section in more detail.
Adding a SmartFolder

In the Order Tracking System main window, the SmartWindow uses three logical pages (0, 1, and 2) to display SmartObjects. Logical pages control which SmartObjects are displayed at a given time. Objects on the default logical page, page 0, are always visible. The SmartFolder uses logical pages 1 and 2 for its tabs. The SmartFolder controls the display of SmartObjects on these pages by toggling their visibility depending on which tab is selected.

**Note:** A SmartFolder can only control the display of SmartObjects. A basic object, such as a text object or a rectangle, must be included in a SmartObject before a SmartFolder can control its display.

To add a SmartFolder:

1. Click the SmartFolder icon in the Object Palette.

2. Position the SmartFolder near the left edge of the Order Line section in the window. The Progress Advisor notifies you that the SmartFolder should be a Page Source for some other SmartObject, and offers to create a Page SmartLink to the container (the SmartWindow):

![Progress Advisor](image)
3. Click **OK**. The folder appears in the design window:

![Folder in design window](image1)

4. Resize and reposition the folder, as shown in the following:

![Resized folder](image2)
5. Click the SmartObject Instance menu, the square icon in the upper-left corner of the SmartFolder, and choose Instance Properties from the pop-up menu:

6. In the SmartFolder Properties dialog box, insert the following tab labels:

<table>
<thead>
<tr>
<th>Tab</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Browser</td>
</tr>
<tr>
<td>2</td>
<td>Detail</td>
</tr>
</tbody>
</table>

The completed dialog box should look like the following:
7. Click OK. The folder now contains two tabs with the labels you entered:

8. Save your work.

Creating page 1 of the SmartFolder

Before you can add objects to the SmartFolder tabs, you must define logical pages for the tabs. In this instance, the Browser tab of the SmartFolder controls logical Page 1. When you add a logical page for a tab, you control which objects appear when that tab is selected.

To define logical pages for the SmartFolder:

1. Click the Procedure Settings icon in the AppBuilder toolbar to open the Procedure Settings dialog box:
2. Choose the **Pages** icon to open the **Pages** dialog box:

![Pages dialog box]

The page numbers begin with Page 0. Page 0 is the default (Main) page. Objects assigned to Page 0 are always displayed, regardless of which folder tab is selected. The objects that you added to the main window so far, such as `dcust`, `dorder`, and the SmartFolder itself, are all assigned to Page 0 by default.

3. Click **Design** to open the **Design Page** dialog box:

![Design Page dialog box]

4. Type **1** in the **Page Number** field and click **OK**. Page 1 is now the current design page and is assigned to the **Browser** tab. Any objects that you add to the SmartWindow now display only when the **Browser** tab is selected.

5. Click the **Start** button on the **Pages** dialog box to open the **Startup on Page** dialog box:

![Startup on Page dialog box]

The value in the **Page Number** field determines which page, if any, displays in addition to Page 0 when the application first runs. In the SportsPro Order Tracking System, you want Page 1 to be the startup page so the SDB displaying the order lines appears at startup.

6. Type **1** in the **Page Number** field and click **OK**. Page 1 now displays at startup, along with the main page.
7. Commit your changes to return to the main window. The **Status Area** of the AppBuilder indicates you are now working in Page 1:

![AppBuilder interface]

The SmartFolder is now configured to control the display of other SmartObjects.

**Adding an SDB to page 1 of the SmartFolder**

Your next task is to add an SDB to Page 1, the **Browser** tab.

To add an SDB to page 1:

1. Click the **SDB** icon from the **Object Palette** to open the **Choose SmartDataBrowser** dialog box:

![Choose SmartDataBrowser dialog box]

2. Select **bord1ne2.w** and click **OK**.

3. Place the SDB in the **Browser** tab. The **Progress Advisor** notifies you that the SDB instance, h_bord1ne2, should be a Data Target for some other SmartObject, and offers to create a Data SmartLink from the SDO instance, h_dord1ne.

4. Select the radio button to create the SmartLink and click **OK**. The **Progress Advisor** notifies you that the SDB instance, h_bord1ne2, should be an Update Source for some other SmartObject, and offers to create an Update SmartLink to the SDO instance, h_dord1ne.

5. Select the radio button to create the SmartLink and click **OK**.
6. Position the SDB in the tab:

Now that you have built the Browser tab, you are ready to build the Detail tab.

**Adding an SDV to page 2 of the SmartFolder**

Before you add the SDV, vordlne, you must reset the current design page.

To specify design page 2 and add an SDV:

1. Double-click the Page Number field in the Status Area of the AppBuilder. The Goto Page dialog box appears:

2. Set the Display Page Number field to 2 and click OK. The main window appears with the Detail tab selected.

3. Choose the SDV icon in the Object Palette to open the Choose SmartDataViewer dialog box.

4. Select vordlne.w and click OK.

5. Place the SDV in the Detail tab. The Progress Advisor notifies you that the SDV instance, h_vordlne, should be a Data Target for some other SmartObject, and offers to add a Data SmartLink from the SDO instance, h_dordlne.
6. Select the radio button to create the SmartLink and click **OK**.

7. Position the **SDV** as shown:

### Adding a Navigation SmartPanel

You now add the Navigation SmartPanel, pnavico, to the Detail tab.

To add a Navigation SmartPanel to the Detail tab:

1. Right-click the **SmartPanel** icon on the **Object Palette**. A pop-up menu listing the available SmartPanels appears:

2. Select **Navigation Panel (Icons)** from the pop-up menu.
3. Add the **SmartPanel** to the left of the **SDV**. The **Progress Advisor** notifies you that the SmartPanel, pnavico, should be a Navigation Source for some other SmartObject, and offers to add a Navigation SmartLink to one of the following SDO instances, h_dcust, h_dorder, or h_dordline:

![Progress Advisor](image1)

4. Select the radio button to create the SmartLink to **h_dordline** and click **OK**:

![SmartPanel](image2)

5. Save your work.

You have now assembled the main window of the SportsPro Order Tracking system using predefined objects. You should have a better understanding of how to use the OpenEdge Studio and AppBuilder to assemble SmartObject applications. Run your application and test the code you created.

If you have not already run the complete sample application, you might want to run it now. Running the application gives you an opportunity to see some of the other capabilities of SmartObjects. For information on running the application, see Chapter 2, “Working with the Sample Application.”

If you would like to work more with SmartObjects, you can complete the exercises in Chapter 4, “Overview of SmartObject Technology.” These exercises show how to fabricate SmartObject masters from templates installed with the product.
Overview of SmartObject Technology

The SportsPro Order Tracking System solves a business problem that a growing distribution business might encounter. Because of the importance of high-level planning in object-based application development, you should spend time designing your application and planning how best to use SmartObjects in the application. This chapter introduces the basics of using SmartObjects. It describes designing an application and using the design to plan the required SmartObjects and SmartLinks, as outlined in the following sections:

- SmartObject technology
- SmartObjects used in the Order Tracking System
Overview of SmartObject Technology

SmartObject technology

The SmartObjects in the main window of the sample application are based on SmartObject templates. Templates provide a starting point for creating master objects. At run time, you run a SmartObject that is an instance of a master. The OpenEdge Studio provides a starter set of SmartObject templates from which to build your own master objects. For example, in the SportsPro Order Tracking System, the SmartObject named bcust is a master based on the SmartDataBrowser (SDB) template.

A significant advantage of SmartObjects is that they are reusable. When you use a SmartObject master in an application, the AppBuilder does not code it directly into the window file. The AppBuilder only writes instructions to create an instance of the master. You can have many applications that use instances of a single master, and you usually use multiple instances of the same master in one application.

SmartObjects are specialized ABL (Advanced Business Language) procedures that run persistently. After a SmartObject performs a specific task, it remains in memory to respond readily to end-user input and instructions from other SmartObjects. The SmartObject’s functionality is encapsulated in a framework of internal procedures (also known as methods). Internal procedures are procedure blocks defined inside the context of an external procedure. Certain internal procedures are common to all SmartObjects, others give each standard SmartObject class their unique characteristics. For more information about internal and external procedures, see OpenEdge Development: AppBuilder, OpenEdge Development: ADM and SmartObjects, and OpenEdge Development: ADM Reference.

Using SmartObjects in applications makes complex applications easy to maintain. Since all instances of a SmartObject master are based on one file, any change you make to that source file applies to every application that uses that master.
SmartObjects used in the Order Tracking System

In Chapter 2, “Working with the Sample Application” and Chapter 3, “Assembling the Order Tracking System,” you examined the design of the SportsPro sample application and learned to use the AppBuilder to create an application from prebuilt SmartObjects. Figure 4–1 shows the main window of the SportsPro Order Tracking System as it appears when you run it.

Figure 4–1 shows the main window of the sample application and identifies some of the SmartObjects in this window.

The main window, worder, is a SmartWindow, which is a type of SmartContainer.

The SmartDataViewer, vcust, displays several fields in a customer’s record.

The SmartDataViewer, vorder, displays several fields in one of the customer’s orders.

The SmartFolder, a direct instance of the SmartObject, folder, manages the viewing and hiding of other SmartObjects organized into logical pages.

The SmartDataBrowser, bordline2, displays a scrolling list of the order lines in the customer’s order.

Some SmartObjects used in the previous Order Tracking System example include:

- **SmartDataObjects (SDOs)** — Sources for Data SmartLinks that target SmartDataViewers and other SDOs
- **SmartFolder** — Source for a Page SmartLink that is targeted to a SmartWindow to control paging in that window
- **SmartDataLinks** — Establish a communication link from SDOs to SDOs and SDVs
- **SmartDataViewer (SDV)** — Displays data from SDOs
- **SmartDataBrowser (SDB)** — Displays data from SDOs and allows a user to scroll through the data
• **SmartPanel** — Contains buttons that control navigation as a source of a Navigation SmartLink through a SDO’s query

• **Navigation SmartLink** — Establishes a navigation SmartLink from a SmartPanel to a SDO

In addition to those objects, the main window uses a SmartContainer. A SmartContainer is a special kind of SmartObject designed to contain other SmartObjects (including possibly other SmartContainers). OpenEdge supplies three types of SmartContainers:

• SmartWindows

• SmartDialogs

• SmartFrames

Each SmartContainer type keeps track of all of the information required to create running instances of the SmartObjects it contains. This information includes the positions of the SmartObjects and how they are linked.

The main window contains different classes of SmartObjects. How a SmartObject functions in an application depends on its class and the SmartLinks defined for the class. SmartLinks are pathways used to send messages among SmartObjects. The set of SmartLinks defined for each class of SmartObject predetermines how that class can interact with other SmartObject classes.

The Progress Advisor that you saw in Chapter 3, “Assembling the Order Tracking System,” uses these sets of possible interactions to make suggestions on how to link the SmartObjects in your application. This eases the task of creating complex applications. However, you get the most benefit from it when you know exactly what pathways you want between the SmartObjects before you begin building your application.
Figure 4–2 shows the SmartObjects and the pathways between them in the main window of the Order Tracking System.

A SmartObject does not know how it relates to other SmartObjects until you create the links. You determine the relationship by using a particular SmartLink type and by specifying which end of the link each object is on (Source or Target). For all SmartLink types, a Source initiates some action or passes some data to one or more Targets. In general, a Source can have numerous Targets simultaneously, but a Target has only one Source at a time. For more information about SmartLinks, see *OpenEdge Development: AppBuilder*.

SmartLinks provide direct communication pathways between pairs of SmartObjects. A single SmartObject can relate to other SmartObjects in a variety of ways. For example, a SDO relates to a SDV differently than it relates to a Navigation SmartPanel. Most SmartObjects support a variety of SmartLink types. Some, like the Navigation SmartPanel, support only one SmartLink type (Navigation).

SmartLinks are directional in nature. Each SmartLink is directed from a source SmartObject to a target SmartObject. There are several types of SmartLinks. For example, a Data SmartLink sends the contents of a database record, while a Navigation SmartLink sends navigation instructions.
The following figures group the SmartObjects shown in Figure 4–2 based on their SmartLinks. The figures describe the types of SmartLinks and the different types of SmartObjects.

Figure 4–3 shows that the SmartFolder, folder, is the source for a Page SmartLink targeted to the SmartWindow, worder. The Page SmartLink tells the main window which logical page to display.

Figure 4–4 shows that the SDO, dcust, is the source for two Data SmartLinks targeted to the SDV, vcust, and the SDO, dorder.
Figure 4–5 shows that the SDO, dorder, is the source for two Data SmartLinks targeted to the SDV, vorder, and the SDO, dordlne.

![SmartLinks for dorder SDO](image)

**Figure 4–5: SmartLinks for dorder SDO**

Figure 4–6 shows that the SDO, dordlne, is the source for two Data SmartLinks targeted to the SDB, bordlne2, and the SDV, vordlne. The SmartPanel, pnavico, is the source for a Navigation SmartLink to the SDO, dordlne.

![SmartLinks for dordlne SDO](image)

**Figure 4–6: SmartLinks for dordlne SDO**

### Naming conventions

The AppBuilder creates structured procedure files, called window files, that are identified by a .w extension. For example, the source filename of the Customer SDB is bcust.w. It has a .r extension in its compiled form (bcust.r) and a prefix of h_ (for handle) when referred to as an instance in memory (h_bcust).

Other files commonly used with the AppBuilder are: include files (.i extension), procedure files (.p extension), and client proxy files (a _cl suffix added to the filename).
Table 4–1 lists the AppBuilder filename conventions.

**Table 4–1: AppBuilder filename conventions**

<table>
<thead>
<tr>
<th>If a filename begins with . . .</th>
<th>Then the file is a . . .</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>SmartDataBrowser (SDB)</td>
</tr>
<tr>
<td>d</td>
<td>SmartDataObject (SDO)</td>
</tr>
<tr>
<td>g</td>
<td>SmartDialog</td>
</tr>
<tr>
<td>p</td>
<td>SmartPanel</td>
</tr>
<tr>
<td>sbo</td>
<td>SmartBusinessObject (SBO)</td>
</tr>
<tr>
<td>smr</td>
<td>Smart[Message]Receiver</td>
</tr>
<tr>
<td>sms</td>
<td>Smart[Message]Sender</td>
</tr>
<tr>
<td>sbb</td>
<td>SmartB2BObject</td>
</tr>
<tr>
<td>sdf</td>
<td>SmartDataField (SDF)</td>
</tr>
<tr>
<td>v</td>
<td>SmartDataViewer (SDV)</td>
</tr>
<tr>
<td>w</td>
<td>SmartWindow</td>
</tr>
</tbody>
</table>
Where to go from here

This book provides you with a practical introduction to the OpenEdge Studio, SmartObject technology, and the basic techniques of AppBuilder application assembly and SmartObject fabrication. You now have the basic information you need to create effective, easily maintainable end-user applications that require little or no manually written ABL code.

However, the OpenEdge Studio is an open system, many features of which are fully customizable. For example, you are not limited to the types of SmartObject templates that are represented in the AppBuilder’s Object Palette. While this starter set of templates is designed to address a wide range of database-oriented application development needs, you might want to create your own SmartObject templates based on the business standards and practices of individual enterprises. The OpenEdge Studio enables you to create custom SmartObject templates and the Cue Cards and Wizards to accompany them. You can then add buttons, directory lists, and file filters for the custom SmartObject templates you create (and for the SmartObject masters that are based on your custom templates).

The OpenEdge Studio enables you to create applications using an assortment of technologies, deploy those applications in different environments, and integrate them with non-OpenEdge applications. Table 4–2 lists some of the possibilities you might want to explore and where you can read about them in the documentation set.

<table>
<thead>
<tr>
<th>Subject</th>
<th>Document</th>
</tr>
</thead>
<tbody>
<tr>
<td>Designing applications with service oriented architectures (SOA)</td>
<td>OpenEdge Getting Started: Application and Integration Services</td>
</tr>
<tr>
<td>Using the AppServer for OpenEdge applications</td>
<td>OpenEdge Application Server: Developing AppServer Applications</td>
</tr>
<tr>
<td>Developing applications for the Web</td>
<td>OpenEdge Getting Started: WebSpeed Essentials</td>
</tr>
<tr>
<td>Integrating with non-Progress ABL applications</td>
<td>OpenEdge Development: Open Client Introduction and Programming</td>
</tr>
<tr>
<td>Developing Web services</td>
<td>OpenEdge Development: Web Services</td>
</tr>
<tr>
<td>Messaging and external applications</td>
<td>OpenEdge Development: Messaging and ESB</td>
</tr>
<tr>
<td>Deploying applications</td>
<td>OpenEdge Deployment: Managing ABL Applications</td>
</tr>
<tr>
<td>Using the OpenEdge RDBMS</td>
<td>OpenEdge Data Management: Database Administration</td>
</tr>
<tr>
<td>Debugging applications</td>
<td>OpenEdge Development: Debugging and Troubleshooting</td>
</tr>
<tr>
<td>Developing repository-based applications</td>
<td>OpenEdge Development: Progress Dynamics Getting Started</td>
</tr>
</tbody>
</table>
### Table 4–2: Information in the documentation set

<table>
<thead>
<tr>
<th>Subject</th>
<th>Document</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic reporting</td>
<td><em>OpenEdge Reporting: Query/Results for Windows</em></td>
</tr>
<tr>
<td>Using SQL with the OpenEdge Studio</td>
<td><em>OpenEdge Data Management: SQL Development</em></td>
</tr>
</tbody>
</table>
Index

A

ActiveX (OCX) controls 3–37
   adding spin boxes 3–38
ActiveX Automation objects
defined 3–37
Advanced Editing Features
   Alias expansion 1–16
   Color coding 1–15
   Embedded color coding 1–16
   SmartPaste 1–16
   Syntax expansion 1–16
   Syntax indenting 1–16
Advanced Properties dialog box 3–28
Alias expansion
   advanced editing features 1–16
AppBuilder
   basic property sheet 3–8
   current object fields 1–10
   customizing Object Palette 1–14
defined 1–4
   File field 1–10
   Label field 1–10
   main window 1–7
   Master field 1–10
   Menu bar 1–8
   Object field 1–10
   Object Palette 1–11
   OCX (ActiveX) field 1–10
   overview 1–7
   Section Editor defined 1–4
   starting 1–7
   status area 1–10

Text field 1–10
Title field 1–10
AppBuilder filename conventions 4–8
AppBuilder main window
   elements 1–8
   Window Properties Window option 3–39
AppBuilder menu bar
   Compile menu 1–8
   Edit menu 1–8
   File menu 1–8
   Help menu 1–8
   Layout menu 1–8
   Options menu 1–8
   Tools menu 1–8
   Window menu 1–8
AppBuilder status area
   File status 1–10
   Lock status 1–10
   Object Palette status 1–10
   Page status 1–10, 3–53
AppBuilder toolbar
   Colors button 1–10
   Development Mode button 1–10
   Edit code button 1–9
   List Objects button 1–9
   New button 1–9
   Object Properties button 1–10
   Open button 1–9
   Print button 1–9
   Procedure Properties button 1–9
   Run button 1–9
   Save button 1–9
   Stop button 1–9
Index–2

Application Compiler
  defined 1–5

Application Debugger
  defined 1–5

B

Browse object
  AppBuilder Object Palette 1–12

Browser tab
  Find Order window 2–37
  in SmartFolder 2–39

Button object
  AppBuilder Object Palette 1–12

Buttons
  adding help text 3–25
  adding images 3–25
  adding ToolTips 3–25

C

Changing orders
  Order Entry/Maintenance window 2–39

Choose Color dialog box 3–14

Choose Control dialog box 3–38

Choose Event dialog box 3–41

Choose Image Files dialog box 3–10

Choose SmartDataViewer dialog box 3–20

Color coding
  advanced editing features 1–15

Colored text labels
  adding 3–40

Colors button
  AppBuilder toolbar 1–10

Combo-Box object
  AppBuilder Object Palette 1–12

Compile menu
  AppBuilder 1–8

Compiling
  sample files 2–7

Connect Database dialog box 2–4

Context-sensitive help
  accessing 1–6

Create Database dialog box 2–3

CSComboBox object
  AppBuilder Object Palette 1–13

CSSpin object
  AppBuilder Object Palette 1–13

Cue Cards 3–7

Current object field
  File field 1–10
  Label field 1–10
  Master field 1–10
  Object field 1–10
  OCX (ActiveX) field 1–10
  Text field 1–10
  Title field 1–10

Current object fields
  AppBuilder 1–10

D

Data Administration tool
  defined 1–4

Data Dictionary
  connecting to Sports2000 database 2–3
  creating a connection 2–3
  defined 1–4

Data Dictionary window 2–4

Database connection
  Connect Database dialog box 2–4

DataView object
  AppBuilder Object Palette 1–13

DB-Fields object
  AppBuilder Object Palette 1–12

Design Page dialog box 3–51

Design page number
  changing 3–53

Designing an application 2–26

Desktop
  menu bar 1–2
  toolbar 1–2

Detail tab
  in SmartFolder 2–39

Development Mode button
  AppBuilder toolbar 1–10

Dialog boxes
  Choose Color 3–14
  Choose Control 3–38
  Choose Event 3–41
  Choose Image Files 3–10
  Choose SmartDataObject 3–19
Choose SmartDataViewer 3–20
Connect Database 2–4
Create Database 2–3
Design Page 3–51
Insert Procedure Call 3–42
List Objects 3–15, 3–27
Multi-Field Mapping 3–33
New 3–7
Pages 3–51
Procedure Settings 3–50
Property Editor 3–39
SmartLinks 3–22
Startup on Page 3–51

Dictionary Startup dialog box 2–3
Display Customer window 2–31
Drag and Drop module 2–14
components 2–15
Dynamic browser
Find Order window 2–37
Dynamic Buffer/Temp Table module 2–12
components 2–13
Dynamic query
in Find Order window 2–37
Dynamic Query module 2–10
components 2–11

E
Edit code button
AppBuilder toolbar 1–9
Edit menu
AppBuilder 1–8
Editor object
AppBuilder Object Palette 1–12
Embedded color coding
advanced editing features 1–16

F
File field
AppBuilder 1–10
File menu
AppBuilder 1–8
File status
AppBuilder status area 1–10
File types
naming conventions 4–8
Fill-In object
AppBuilder Object Palette 1–12
Find Customer window 2–29
Find Order window 2–37
Browser section 2–37
dynamic browser 2–37
order filter section 2–37
search query 2–37
Selected Fields Displayed 2–37
SmartFilter 2–37
Frame object
AppBuilder Object Palette 1–12

H
Help file
adding 3–27
Help menu
AppBuilder 1–8
Help text
adding to buttons 3–25

I
Image object
AppBuilder Object Palette 1–12
Images
adding to buttons 3–25
adding to SmartWindow 3–9
property sheet 3–10
Insert Procedure Call dialog box 3–42
Instance attributes
SmartFolder 3–49
Inventory Trans dialog box 2–41
Item Maintenance window 2–40
Item Selection window 2–35
SmartSelect Browser 2–35

L
Label field
AppBuilder 1–10
Layout menu
AppBuilder 1–8
List Objects button
AppBuilder toolbar 1–9
List Objects dialog box 3–15
Lock status
AppBuilder status area 1–10

Logical pages
adding to SmartFolders 3–50
Page 0 3–51
SmartDataViewer 2–39

M
Main window
Order Line section 2–39
Maintain Customer window 2–31
SmartDataField 2–31
SmartToolBar 2–31

Master field
AppBuilder 1–10

Menu bar
AppBuilder 1–8
Desktop 1–2

Modifying OCX spin boxes 3–39

Multi-Field Mapping dialog box 3–33

N
Named-events module 2–20
Navigation SmartPanel
adding 3–54
Item Maintenance window 2–40
Order section 2–34

New button
AppBuilder toolbar 1–9
New dialog box 3–7

O
Object Component Extension (OCX)
defined 3–37

Object field
AppBuilder 1–10

Object Palette
Button icon 3–23
controlling with mouse 1–15
customizing 1–14, 4–9
defined 1–11
OCX icon 3–38
Options 1–11
Rectangle icon 3–12
setting characteristics 1–11
SmartDataBrowser icon 3–52
SmartDataObject icon 3–19, 3–34

Object Palette objects
Browse 1–12
Button 1–12
Combo-Box 1–12
CSSComboBox 1–13
CSSpin 1–13
DataViewr 1–13
DB-Fields 1–12
Editor 1–12
Fill-In 1–12
Frame 1–12
Image 1–12
OCX 1–13
Pointer 1–12
PSTimer 1–13
Query 1–12
Radio-Set 1–12
Rectangle 1–12
Selection-List 1–12
Slider 1–12
SmartB2BOBJECT 1–14
SmartBusinessObject 1–14
SmartConsumer 1–14
SmartContainer 1–14
SmartDataBrowser 1–13
SmartDataField 1–13
SmartDataObject 1–13
SmartDataViewer 1–13
SmartFilter 1–13
SmartFolder 1–13
SmartLOBField 1–13
SmartObject 1–13
SmartPanel 1–13
SmartProducer 1–14
SmartReceiver 1–14
SmartRouter 1–14
SmartSender 1–14
SmartToolBar 1–13
Text 1–12
Toggle-Box 1–12

Object Palette status
AppBuilder status area 1–10

Object Properties button
AppBuilder toolbar 1–10
creating SmartWindows 3–8

OCX (ActiveX) field
AppBuilder 1–10

OCX controls
adding spin boxes 3–38
licensed by Progress 3–38
third-party objects 3–38

OCX object
AppBuilder Object Palette 1–13
Online Help
accessing context-sensitive help 1–6
AppBuilder 1–8
defined 1–6

Open button
AppBuilder toolbar 1–9

OpenEdge Studio
Desktop 1–2
starting 1–2
toolset 1–4

Options menu
AppBuilder 1–8

Order Entry/Maintenance window
changing orders 2–39

Order filter section
Find Order window 2–37

Order Line section
SportsPro Order Tracking System 2–39

Order section
adding buttons 3–36
Navigation panel 2–34
SmartDataViewer 2–34

P

Page status
AppBuilder status area 1–10

Pages dialog box 3–51

Partially assembled application
viewing 3–44

Pointer object
AppBuilder Object Palette 1–12

Print button
AppBuilder toolbar 1–9

PRO*Tools
defined 1–4

Procedure Editor
defined 1–4

Procedure files 4–7

Procedure Properties button
AppBuilder toolbar 1–9

Procedure Settings dialog box 3–50

Progress Advisor 3–2

property 3–21

Property sheets
Image 3–10
rectangle 3–12
SmartDataViewer 3–21
SmartWindow 3–8, 3–16, 3–27
text label 3–14

PSTimer object
AppBuilder Object Palette 1–13

Publish and Subscribe module 2–20
components 2–21

Q

Query object
AppBuilder Object Palette 1–12

Querying
order information 3–30

R

Radio-Set object
AppBuilder Object Palette 1–12

Rectangle object
AppBuilder Object Palette 1–12

Rectangle property sheet 3–12

Rectangles
adding to SmartWindow 3–12

Resizable Dynamic Browser module 2–15
components 2–16

Results
defined 1–4

Run button
AppBuilder toolbar 1–9
running sample application 2–8

Running tools
instances limit 1–3

S

Save button
AppBuilder toolbar 1–9

Search criteria 2–37

Search query
Find Order window 2–37
Section Editor
adding trigger code 3–28
New button 3–41

Selection-List object
AppBuilder Object Palette 1–12

Set Application Language window 2–42

Slider object
AppBuilder Object Palette 1–12

SmartB2B and SonicMQ module 2–22
components 2–24

SmartB2BObject object
AppBuilder Object Palette 1–14

SmartBusinessObject object
AppBuilder Object Palette 1–14

SmartConsumer object
AppBuilder Object Palette 1–14

SmartContainer object
AppBuilder Object Palette 1–14

SmartContainers
definition 4–4

SmartDataBrowser object
AppBuilder Object Palette 1–13

SmartDataBrowsers
adding to SmartWindow 3–52
Inventory Trans dialog box 2–41
Item Maintenance window 2–40
Item Selection window 2–35

SmartDataField module 2–19
components 2–20

SmartDataField object
AppBuilder Object Palette 1–13

SmartDataObject module 2–17
components 2–18

SmartDataObject object
AppBuilder Object Palette 1–13

SmartDataObjects
adding to Order Section 3–32
adding to SmartWindow 3–18

SmartDataViewer object
AppBuilder Object Palette 1–13

SmartDataViewers
adding to Customer Section 3–20
adding to SmartWindow 3–20
Display Customer window 2–20
Item Maintenance window 2–31
logical pages 2–39

Maintain Customer window 2–31
Order Entry/Maintenance window 2–35
Order Line information 2–39
Order Line section 2–39
Order section 2–34, 3–35
property sheet 3–21

SmartFilter
Find Order window 2–37

SmartFilter object
AppBuilder Object Palette 1–13

SmartFolder Attributes dialog box 3–49

SmartFolder object
AppBuilder Object Palette 1–13

SmartFolders 3–50
adding 3–52
adding SmartObjects to pages 3–52
adding to SmartWindow 3–47
Browse tab 2–39
choosing startup page 3–51
creating pages 3–50
detail tab 2–39
instance attributes 3–49
logical pages 3–50

SmartLinks 3–2
creating links between 3–21
data 4–5
defined 4–4
navigation 4–5
page 4–6
types 4–5

SmartLinks dialog box 3–22

SmartLOBField object
AppBuilder Object Palette 1–13

SmartObject applications 2–26

SmartObject object
AppBuilder Object Palette 1–13

SmartObjects
adding to SmartFolder page 3–52
controlling displays 3–47
defined 4–2
internal procedures 4–2
running persistently 4–2
SmartContainers 4–2
SmartLinks 4–4
templates 4–2
used in main window 4–3

SmartPanel object
AppBuilder Object Palette 1–13

SmartPanels
adding to SmartFolder page 3–54
SmartPaste
   advanced editing features 1–16
SmartProducer object
   AppBuilder Object Palette 1–14
SmartReceiver object
   AppBuilder Object Palette 1–14
SmartRouter object
   AppBuilder Object Palette 1–14
SmartSelect Browser 2–35
SmartSender object
   AppBuilder Object Palette 1–14
SmartToolBar object
   AppBuilder Object Palette 1–13
SmartWindows
   adding images 3–9
   adding OCX controls 3–38
   adding rectangles 3–12
   adding SmartFolder 3–47
   adding SmartObjects 3–17
   adding spin boxes 3–38
   adding text labels 3–13
   assembling main window 3–5
   creating 3–6
   property sheet 3–8, 3–16, 3–27
   saving 3–15
Spin boxes
   adding to SmartWindow 3–38
   OCX control 3–38
Sports2000 database 2–3
   connecting from Data Dictionary 2–3
SportsPro Order Tracking System
   2–9
   business scenario 2–26
   components 4–7
   customer maintenance components 2–32
   Customer section 2–29
   find order components 2–38
   inventory transfer components 2–41
   main window components 2–27
   menu items 2–40
   order entry/maintenance components 2–35
   Order Line section 2–39
   set application language components 2–42
SportsPro Sample Application
   Drag and Drop module 2–14
   Dynamic Buffer/Temp Table module 2–12
   Dynamic Query module 2–10
   main window components 2–9
   Publish and Subscribe module 2–20
Resizable Dynamic Browser module 2–15
   running completed application 2–8
SmartB2B and SonicMQ module 2–22
SmartDataField module 2–19
SmartDataObject module 2–17
   using PRO*Tools to run 2–8
Starting a tool 1–2
Startup on Page dialog box 3–51
Status area
   AppBuilder 1–10
   logical page 3–52
Stop button
   AppBuilder toolbar 1–9
Syntax expansion
   advanced editing features 1–16
Syntax indenting
   advanced editing features 1–16
T
   Tabs
      in SmartFolders 2–39
Text field
   AppBuilder 1–10
Text labels
   adding to SmartWindows 3–13
   selecting colors 3–14
Text object
   AppBuilder Object Palette 1–12
Third-party objects
   OCS Controls 3–38
Title field
   AppBuilder 1–10
Toggle-Box object
   AppBuilder Object Palette 1–12
Toolbar
   Desktop 1–2
Tools
   AppBuilder 1–4
   Application Compiler 1–5
   Application Debugger 1–5
   Data Administration 1–4
   Data Dictionary 1–4
   launching from another tool 1–3
   launching from Desktop 1–2
   PRO*Tools 1–4
   Procedure Editor 1–4
   Results 1–4
Translation Manager 1–4
Visual Translator 1–5
Web Tools 1–4

Tools menu
AppBuilder 1–8
list of tools 1–4
starting another tool 1–3

Toolset
list of tools 1–4

ToolTips 1–2
adding to buttons 3–25

Translation Manager
defined 1–4

Trigger code
adding to buttons 3–28
adding to spin box 3–41
adding with Section Editor 3–28

Update panel 2–34

Update SmartPanel
in Item Maintenance window 2–40

User-interface triggers
fetching first record 3–43
fetching last record 3–43
fetching next record 3–43
fetching previous record 3–41

V

Viewing order information 3–30

Visual Translator
defined 1–5

W

Web Tools
defined 1–4

Window files 4–7

Window menu
AppBuilder 1–8