OPENEDGE 10

OpenEdge® Deployment:
WebClient® Applications
# Contents

Preface ................................................................... Preface–1

1. **Overview** ......................................................... 1–1
   Differences between WebClient and the standard ABL client ................. 1–2
   Common WebClient scenarios ........................................... 1–3
   ASP scenario .............................................................. 1–3
   IT department scenario ............................................... 1–3
   WebClient files and software ........................................... 1–4
   WebClient versus WebClient applications .......................... 1–4
   Personnel involved in WebClient deployments ........................ 1–5
   WebClient installation types .......................................... 1–6
   Administrator installations ............................................ 1–6
   Personal installations .................................................. 1–6
   Installation precedence ............................................... 1–7
   Requirements for using WebClient ................................... 1–10
   WebClient architecture ................................................ 1–11
   WebClient Application Assembler .................................. 1–11
   WebClient Deployment Packager ................................... 1–12
   Using IntelliStream technology ...................................... 1–12
   Using non-IntelliStream technology .................................. 1–14
   Using IntelliStream with non-IntelliStream technologies ............... 1–15
   Running the application ............................................... 1–16
   Single sign-on ................................................................ 1–16
   Server sharing ............................................................ 1–17
   Application requirements .............................................. 1–18
   Access databases only through an AppServer ........................ 1–18
   Modularize code by function ......................................... 1–18
   Choosing which installation types to support ........................ 1–19

2. **Designing Your End User’s Experience** .................... 2–1
   Installation process ..................................................... 2–2
   Microsoft Vista installations ......................................... 2–2
   WebClient application installation .................................. 2–4
   IntelliStream .............................................................. 2–5
   External installer ........................................................ 2–6
<table>
<thead>
<tr>
<th>Contents</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Updates for your application</td>
<td>2–7</td>
</tr>
<tr>
<td>Uninstalling your application</td>
<td>2–8</td>
</tr>
<tr>
<td>Starting WebClient applications</td>
<td>2–9</td>
</tr>
<tr>
<td>Providing documentation</td>
<td>2–10</td>
</tr>
<tr>
<td>3. Hosting the WebClient Deployment</td>
<td>3–1</td>
</tr>
<tr>
<td>Hosting application configuration files</td>
<td>3–2</td>
</tr>
<tr>
<td>Hosting codebases</td>
<td>3–3</td>
</tr>
<tr>
<td>Hosting the codebase on an internet-based server</td>
<td>3–3</td>
</tr>
<tr>
<td>Hosting the codebase on an AppServer</td>
<td>3–3</td>
</tr>
<tr>
<td>4. Designing Security</td>
<td>4–1</td>
</tr>
<tr>
<td>Digitally signing files</td>
<td>4–2</td>
</tr>
<tr>
<td>How WebClient uses digital signatures</td>
<td>4–2</td>
</tr>
<tr>
<td>Protecting servers with user IDs and passwords</td>
<td>4–5</td>
</tr>
<tr>
<td>Prompting the end user for authentication information</td>
<td>4–5</td>
</tr>
<tr>
<td>Single sign-on and security caching</td>
<td>4–7</td>
</tr>
<tr>
<td>Using WebClient logon dialogs in your applications</td>
<td>4–8</td>
</tr>
<tr>
<td>Choosing secure communication protocols</td>
<td>4–9</td>
</tr>
<tr>
<td>5. Developing the Application</td>
<td>5–1</td>
</tr>
<tr>
<td>Using the CODEBASE-LOCATOR handle and its attributes</td>
<td>5–2</td>
</tr>
<tr>
<td>Basic attributes</td>
<td>5–2</td>
</tr>
<tr>
<td>Security-cache attributes</td>
<td>5–3</td>
</tr>
<tr>
<td>Additional attribute</td>
<td>5–4</td>
</tr>
<tr>
<td>Checking application registry access</td>
<td>5–5</td>
</tr>
<tr>
<td>Managing AppServer connections</td>
<td>5–6</td>
</tr>
<tr>
<td>Implementing single sign-on</td>
<td>5–7</td>
</tr>
<tr>
<td>Using WebClient’s authentication dialogs</td>
<td>5–8</td>
</tr>
<tr>
<td>Displaying the AIA authentication dialog</td>
<td>5–8</td>
</tr>
<tr>
<td>Displaying the AppServer authentication dialog</td>
<td>5–9</td>
</tr>
<tr>
<td>Displaying the AIA and AppServer authentication dialog</td>
<td>5–10</td>
</tr>
<tr>
<td>Using URLs in PROPATH for image file downloads</td>
<td>5–11</td>
</tr>
<tr>
<td>Using the SEARCH function</td>
<td>5–11</td>
</tr>
<tr>
<td>Using the LOAD–IMAGE( ) method</td>
<td>5–11</td>
</tr>
<tr>
<td>Handling errors</td>
<td>5–12</td>
</tr>
<tr>
<td>Compiling the application</td>
<td>5–14</td>
</tr>
<tr>
<td>6. Deploying an Application</td>
<td>6–1</td>
</tr>
<tr>
<td>Preparing to use the WebClient Application Assembler</td>
<td>6–2</td>
</tr>
<tr>
<td>Required input files</td>
<td>6–2</td>
</tr>
<tr>
<td>Installation type considerations</td>
<td>6–3</td>
</tr>
<tr>
<td>Administrator installations</td>
<td>6–3</td>
</tr>
<tr>
<td>Personal installations</td>
<td>6–4</td>
</tr>
<tr>
<td>Choosing an installation method</td>
<td>6–6</td>
</tr>
<tr>
<td>IntelliStream</td>
<td>6–6</td>
</tr>
<tr>
<td>External installer</td>
<td>6–7</td>
</tr>
<tr>
<td>Using a combination of IntelliStream and an external installer</td>
<td>6–11</td>
</tr>
<tr>
<td>Defining applications in WebClient Application Assembler</td>
<td>6–14</td>
</tr>
<tr>
<td>General tab</td>
<td>6–15</td>
</tr>
<tr>
<td>Component tab</td>
<td>6–16</td>
</tr>
<tr>
<td>Options tab</td>
<td>6–17</td>
</tr>
<tr>
<td>WebClient tab</td>
<td>6–20</td>
</tr>
<tr>
<td>Files generated by the WebClient Application Assembler</td>
<td>6–21</td>
</tr>
</tbody>
</table>
### Customizing deployment configurations
- Distribution of the Deployment Packager: 6–23
- Changes allowed through the Deployment Packager: 6–24
- Defining a custom deployment configuration: 6–24
- Validating the custom deployment configuration: 6–26
- Generating the _prowcapc_ file: 6–26

### Hosting the application on a server
- Hosting the configuration file: 6–28
- Hosting your application component files: 6–29
- Hosting your external installer: 6–29
- Hosting on a Web server: 6–29

### Installing WebClient
- Accessing the WebClient installation files: 6–33
- Customizing the WebClient installation: 6–34

### Bootstrapping a WebClient application
- WebClient application updates: 6–42
- Updating procedure libraries: 6–43

### WebClient and OpenEdge GUI for .NET
- .NET Framework: 6–44
- .NET assemblies: 6–45
- Other files: 6–46

### 7. Your End Users' Experience
- Preparing documentation for end users: 7–2
- WebClient Application Manager: 7–4
  - Using the WebClient Application Manager: 7–4
  - WebClient tab: 7–5
  - Applications tab: 7–6
  - Edit dialog: 7–6

### A. Deploying the Sample Application with IntelliStream
- Preparing to run the sample application: A–2
- Preparing the Web server: A–3
  - Setting up the Java Servlet Engine and the AppServer Internet Adapter: A–3
  - Setting up MIME types for the application configuration file: A–3
  - Setting up the WebClient install and bootstrap mechanism: A–4
- Setting up the application on the AppServer machine: A–5
- Setting up the application to launch from the Web server: A–8
- Installing and running the application on the client machine: A–9

### B. Deploying the Sample Application Without IntelliStream
- Preparing to run the sample application: B–2
- Preparing the Web server: B–3
  - Setting up the Java Servlet Engine and the AppServer Internet Adapter: B–3
  - Setting up MIME types for the application configuration file: B–3
- Setting up the application on the AppServer machine: B–4
- Setting up the application on the Web server machine: B–6
- Installing and running the application on the client machine: B–8

### Index
- Index–1
### Tables

<table>
<thead>
<tr>
<th>Table 1–1:</th>
<th>Comparing WebClient to the standard ABL client</th>
<th>1–2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 5–1:</td>
<td>Basic attributes of the CODEBASE-LOCATOR handle</td>
<td>5–2</td>
</tr>
<tr>
<td>Table 5–2:</td>
<td>Security-cache attributes of the CODEBASE-LOCATOR handle</td>
<td>5–3</td>
</tr>
<tr>
<td>Table 5–3:</td>
<td>Additional attribute of the CODEBASE-LOCATOR handle</td>
<td>5–4</td>
</tr>
<tr>
<td>Table 6–1:</td>
<td>Administrator installation required permissions</td>
<td>6–3</td>
</tr>
<tr>
<td>Table 6–2:</td>
<td>Personal installation default values</td>
<td>6–4</td>
</tr>
<tr>
<td>Table 6–3:</td>
<td>Registry entries made by WebClient</td>
<td>6–8</td>
</tr>
<tr>
<td>Table 6–4:</td>
<td>Vendor and application name registry key values</td>
<td>6–10</td>
</tr>
<tr>
<td>Table 6–5:</td>
<td>Configuration file registry key entries</td>
<td>6–10</td>
</tr>
<tr>
<td>Table 6–6:</td>
<td>WebClient applications registry key value</td>
<td>6–11</td>
</tr>
<tr>
<td>Table 6–7:</td>
<td>Component registry key entry</td>
<td>6–12</td>
</tr>
</tbody>
</table>
Figures

Figure 1–1: Launching an Administrator instance of a WebClient application . . . . 1–8
Figure 1–2: Launching a personal instance of a WebClient application ......... 1–9
Figure 6–1: Bootstrapping process ...................................................... 6–38
Figure 7–1: General tab with WebClient Application Assembler info .......... 7–7
Figure 7–2: General tab without WebClient Application Assembler info ........ 7–7
Procedures

Sample bootstrap.htm file ................................................................. 6–39
This Preface contains the following sections:

- Purpose
- Audience
- Organization
- Using this manual
- Typographical conventions
- Example procedures
- OpenEdge messages
- Third party acknowledgements
Purpose

This manual describes the architecture and options for enabling a standard OpenEdge® application for deployment over the Web using the OpenEdgeWebClient™. It provides guidance on WebClient implementations that are typical in the following application scenarios:

- The application Service Provider (ASP) deploying a rental application over the Internet.
- An Information Technology (IT) department deploying corporate applications over an intranet.

Audience

The audience for this manual includes:

- OpenEdge developers who build, deploy, and manage applications across the Internet, intra- and extra-nets, multi-tier, client/server, and host/terminal computing environments.
- Any service provider who deploys a WebClient application developed by someone else.

Organization

Chapter 1, “Overview”

Describes the architecture, requirements, and typical application scenarios for using the WebClient to implement OpenEdge on the Web, including an overview of typical implementation steps.

Chapter 2, “Designing Your End User’s Experience”

Describes the available delivery options for deploying a WebClient application, including recommended combinations.

Chapter 3, “Hosting the WebClient Deployment”

Describes and compares the options for configuring the deployment environment for a WebClient application, including information on supported servers and server software.

Chapter 4, “Designing Security”

Provides an overview of security concepts and describes how the WebClient architecture supports security for typical application scenarios.

Chapter 5, “Developing the Application”

Describes how to use ABL to implement WebClient application design decisions and requirements.
Chapter 6, “Deploying an Application”

Provides a guide to the tasks you must complete and the tools you must use to implement your delivery design and deploy a WebClient application in your deployment environment.

Chapter 7, “Your End Users’ Experience”

Provides information that your application users might need to know to install and run your application, and to help you troubleshoot any WebClient applications running on their system.

Appendix A, “Deploying the Sample Application with IntelliStream”

Describes how to deploy the SprotsPro sample application using WebClient with IntelliStream™.

Appendix B, “Deploying the Sample Application Without IntelliStream”

Describes how to deploy the SportsPro sample application using WebClient without IntelliStream.

Using this manual

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 Overview page on PSDN: http://communities.progress.com/pcom/docs/DOC-16074.

References to ABL compiler and run-time features

ABL is both a compiled and an interpreted language that executes in a run-time engine. The documentation refers to this run-time engine as the ABL Virtual Machine (AVM). When the 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 the 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, built-in 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 character appears, this is a reference to the CHARACTER, LONGCHAR, or CLOB 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 built-in 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><strong>Bold</strong></td>
<td>Bold typeface indicates commands or characters the user types, provides emphasis, or the names of user interface elements.</td>
</tr>
<tr>
<td><em>Italic</em></td>
<td>Italic typeface indicates the title of a document, 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 plus sign 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>
<tr>
<td><strong>Syntax:</strong></td>
<td></td>
</tr>
<tr>
<td>Fixed width</td>
<td>A fixed-width font is used in syntax statements, code examples, system output, and filenames.</td>
</tr>
<tr>
<td>Fixed-width italics</td>
<td>Fixed-width italics indicate variables in syntax statements.</td>
</tr>
<tr>
<td>Fixed-width bold</td>
<td>Fixed-width bold indicates variables with special emphasis.</td>
</tr>
<tr>
<td>UPPERCASE fixed width</td>
<td>Uppercase words are ABL keywords. Although these are always shown in uppercase, you can type them in either uppercase or lowercase in a procedure.</td>
</tr>
</tbody>
</table>
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 located in the doc_samples directory on the OpenEdge Product DVD.
- The OpenEdge Product Documentation Overview page on PSDN:

  http://communities.progress.com/pcom/docs/DOC-16074

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 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**

In 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 type 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](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/](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](mailto: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 (dmsctof.c; 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 Apache Software Foundation (http://www.apache.org/). Copyright © 1999 CERN - European Organization for Nuclear Research. Permission to use, copy, modify, distribute and sell this software and its documentation for any purpose is hereby granted without fee, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation. CERN makes no representations about the suitability of this software for any purpose. It is provided "as is" without expressed or implied warranty.

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.
WebClient™ lets you, the application deployer, take a Web-enabled OpenEdge application, deploy and configure its back-end components as desired, package its front-end components, and host these packages on a server. Similarly, you can package updates to the application front end and place them on a server as well. The end user, typically using a Web browser, can request to run the application. In response, WebClient downloads the initial front-end package, installs and runs the application, downloads update packages, and installs the updates—all with a high degree of transparency.

This chapter contains the following sections:

- Differences between WebClient and the standard ABL client
- Common WebClient scenarios
- WebClient files and software
- Personnel involved in WebClient deployments
- WebClient installation types
- Requirements for using WebClient
- WebClient architecture
- Running the application
- Application requirements
## Differences between WebClient and the standard ABL client

Table 1–1 compares WebClient and the standard ABL client.

### Table 1–1: Comparing WebClient to the standard ABL client

<table>
<thead>
<tr>
<th>Feature</th>
<th>WebClient</th>
<th>Standard ABL client</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execution</td>
<td>Contains no compiler; runs as a reduced-size executable.</td>
<td>Compiles source and runs as a complete ABL client executable.</td>
</tr>
<tr>
<td>Data access</td>
<td>Supports temp-table access, but does not support direct access to the OpenEdge® database or DataServers; relies on a connected AppServer™ to provide indirect access to all databases and DataServers.</td>
<td>Supports direct access to all data objects, as well as indirect access to databases and DataServers using a connected AppServer.</td>
</tr>
<tr>
<td>Customization</td>
<td>Cannot be customized using OEBuild.</td>
<td>Can be customized using OEBuild.</td>
</tr>
<tr>
<td>User interface</td>
<td>Supports only the graphical user interface (GUI) and batch modes in Windows.</td>
<td>Supports all ABL user-interface modes.</td>
</tr>
<tr>
<td>Method of launch</td>
<td>Is typically launched by accessing a Web page or a shortcut that runs the WebClient Initializer, which performs certain housekeeping tasks and automatically executes WebClient on the end user’s behalf. The ABL procedures that the application comprises do not have to be installed on the end user’s machine prior to run time.</td>
<td>Is typically launched by executing a script or shortcut you provide that executes the ABL client on the end user’s behalf. The ABL procedures that the application comprises must be installed on the end user’s machine prior to run time.</td>
</tr>
</tbody>
</table>
Common WebClient scenarios

WebClient is specifically designed for:

- Application service providers (ASPs) renting out applications over an extranet.
- Corporate information technology (IT) departments deploying applications internally over a corporate intranet.

ASP scenario

As an ASP, you rent applications to end users, who install and run the applications over the Web or an extranet. Your end users might perform tasks like the following:

- **Identifying the application** — End users identify the application on your Web site as a likely business solution through a Web search.

- **Evaluating the application** — End users sample the application through a live demonstration, installed and executed directly on their systems over the Web. You make this option available by providing the application in a demo mode, complete with online documentation and sample data.

- **Subscribing to the application** — You allow end users access to a measured level of application functionality based on a price-driven subscription agreement. You maintain and administer the application’s back end (its servers, databases, and so on).

- **Downloading, installing, and launching the application** — Your Web site grants end users access to the application, including the purchased level of functionality controlled by a version-based license. This allows them to download and install the application directly from a server, complete with all supporting OpenEdge components. End users then run the application, which access the business logic on your back-end server.

IT department scenario

As an IT department, you install and run your application entirely over a corporate intranet. You or your end users might perform tasks similar to the following:

- **Identifying the application** — You search the Web to identify an application to deploy via WebClient.

- **Evaluating the application** — Your end users sample the application through a live demonstration, installed and executed directly on their systems over the Internet.

- **Purchasing the application** — You buy the application.

- **Deploying the application** — You deploy the application on a local machine.

- **Downloading, installing, and launching the application** — Your end users access the local machine to download, install, run, and update the application, as in the ASP model.
WebClient files and software

The following files and software support your WebClient application deployment:

- **WebClient executable** *(prowc.exe)* — The WebClient main executable, responsible for running the application on the user’s machine.

- **WebClient Initializer** *(prowcini.exe)* — The program responsible for coordinating the installation and execution of a WebClient application. The Initializer is installed with WebClient.

- **WebClient Application Assembler** *(prowcappmgr.exe)* — A utility used to define the application to WebClient and to package the front end of the application for downloading.

- **WebClient Deployment Packager** *(contained within prowcappmgr.exe)* — A utility used to modify the deployment configuration for an application.

- **WebClient Application Manager** *(prowcam.exe)* — A utility for customizing and troubleshooting your WebClient applications on the end user’s machine. It is typically used when the end user talks with your technical support staff.

- **Application Configuration** *(.prowcapp)* file — The configuration file created by the WebClient Application Assembler and downloaded to the end user’s machine at the initial download and each time the end user starts the application. This file provides WebClient with all the information it needs to know about the application.

- **WebClient project** *(.wcp)* file — The file that contains the WebClient application install definition for the latest version of an application.

**Note:** In Windows Explorer, if you double-click on a project file, the WebClient Application Assembler starts and opens that project definition.

- **Microsoft Cabinet** *(.cab)* files — Files into which the procedure files, image files, etc., that make up the front end of the application are placed to prepare them for downloading to the end user’s machine.

- **Microsoft Authenticode™ Technology** — The software that creates and validates digital signatures.

- **Test-certificate generator** *(maketestcert.bat)* — A batch file that simplifies creating test public-key certificates used for testing digitally signed applications.

- **WebClient application** — The application to be deployed through WebClient.

WebClient versus WebClient applications

The WebClient itself is separate from the applications that you deploy with it. When used alone, **WebClient** refers to the WebClient executable and the rest of the infrastructure on which a deployed application runs. **WebClient application** refers to the deployed application itself.
Personnel involved in WebClient deployments

Using WebClient typically involves the following people:

- **Application deployer** — The person who reads the WebClient documentation and uses WebClient to deploy the application. This person might also serve as the application designer, application developer, and system administrator.

- **End user** — The person who uses the application set up by the application deployer.

- **Out-hoster** — A third party who hosts the application on their Web site. An out-hoster might be used when the Web site of the application deployer’s company cannot host the application.

**Note:** Since out-hosters typically lack knowledge of WebClient and OpenEdge, they must work closely with the application deployer.
Overview

WebClient installation types

WebClient installations come in two types: Administrator installations and personal installations. The two types need different permissions and write files to different locations. Application deployers must decide which installations work best with their circumstances.

Administrator installations

The Administrator installation works like the WebClient installations of previous releases. An Administrator installation creates a single installed copy of a WebClient application on a machine to which all end users on that machine have access. To accomplish this, the install writes the WebClient and WebClient application information to areas that are common to all users of that machine, such as the HKEY_LOCAL_MACHINE (HKLM) registry key and the C:\Program Files directory. Using these shared areas requires write permissions that, by default, are restricted to users with Administrator privileges. After installation, you also need these privileges to handle application updates and to download “As Needed” components.

For security reasons, many companies do not want to grant everyone Administrator privileges. To use a WebClient application in that case, administrators have to install the application and then grant limited write privileges to the application users. For more information on this, see the “Granting write privileges” section on page 6–3.

An administrator controls this type of installation throughout its life cycle. The administrator must install the application, grant permissions to end users, and perhaps set up batch jobs to handle application updates.

Personal installations

A personal installation creates a separate copy of the WebClient application for each user. The installation uses the HKEY_CURRENT_USER (HKCU) registry key and the user’s own application data directory, such as C:\Documents and Settings\user-name\Application Data.

When the WebClient installer determines that an end user cannot perform an Administrator install, the installer switches to performing a personal install. The Welcome screen includes the following message:

Note: You do not have Administrator privileges. You can only install WebClient as a personal instance. To make WebClient available to all users, cancel this install and have a user with Administrator privileges install WebClient.

When the installation completes, the final screen contains the following message reminding the user that they installed a personal instance of WebClient:

Note: You have installed a personal instance of WebClient. If other users need access to WebClient, uninstall this instance and have a user with Administrator privileges install WebClient.
An administrator does not control this type of installation. If the end users have access to the WebClient Initializer, they can each decide to install a personal copy of the application on their machine.

**Note:** Some IntelliStream System Tasks require Administrator privileges to install. If your application makes use of these tasks, users cannot install personal instances of your WebClient application. For more information, see the “Administrator installations” section on page 6–3.

Only the user who installs the personal instance has access to it. For example, if an Administrator ran the WebClient Application Manager, he would not see any of the personal instances that users had installed on the machine.

**Installation precedence**

Progress Software Corporation recommends that you do not have both installation types of the same WebClient application on a single machine. If several users on the same machine use a WebClient application, a single Administrator installation uses less resources. You should uninstall any personal installations of an application if an Administrator installation exists.

However, circumstances might cause a machine to host a mix of Administrator and personal instances of WebClient applications. As a result, a user might attempt to use a personal instance of a particular application that they installed before the Administrator instance was installed. You might also have several different WebClient applications installed on the same machine. When there are multiple WebClient installations on a single machine, an Administrator installation generally has precedence over a personal installation.

Figure 1–1 and Figure 1–2 show the logic process by which WebClient Initializer chooses which instances of the WebClient and the WebClient application to start. The first thing that the Initializer checks is if there is an Administrator instance of WebClient application available. If there is an Administrator instance of the WebClient application, it runs through the process shown in Figure 1–1. If not, it runs through the process shown in Figure 1–2.

Note that, if there is no appropriate version of the WebClient available, the process does not upgrade an existing instance. Instead, it installs a completely new instance. This preserves the already installed instances for any other WebClient application that might need those particular versions.

**Note:** For Administrator installations, an “Administrator user” is any user who has sufficient privileges to perform installations or upgrades.

When an end user does not have sufficient privileges to update an Administrator instance of a WebClient application, the Initializer launches the last installed version of the application. However, the end user might see error messages when necessary upgrades cannot be made.
If the Initializer does not find an appropriate version of the WebClient application, the result depends on how the application deployer designs the deployment. If the deployer specified an upgrade path for the currently installed version, the Initializer launches the upgrade. If the deployer did not specify an upgrade path, then the Initializer installs a new personal instance of the WebClient application.

Figure 1–1: Launching an Administrator instance of a WebClient application

Note: Figure 1–1 assumes that you have already performed the initial installation of the WebClient application. The asterisk (*) indicates that the user might see errors when they cannot make necessary upgrades.
When the Initializer does not find an appropriate version of the WebClient application, it checks whether or not the deployer has specified an upgrade path for the version of the existing instance. If so, it upgrades the existing instance. Otherwise, it installs a new instance of the WebClient application. Administrators might need to uninstall superseded instances to free up disk space.

Note that even for a personal installation of a WebClient application, the Initializer first checks if there is an Administrator instance of WebClient from some other installation with an acceptable version. If it finds no appropriate Administrator instance of WebClient, it switches to looking for personal instances of WebClient.

![Diagram of WebClient installation types]

**Figure 1–2:** Launching a personal instance of a WebClient application
Requirements for using WebClient

To successfully install and use a WebClient application, the end user’s machine requires the following:

- Microsoft Windows XP or Microsoft Vista operating systems

  **Note:** Installations in Microsoft Vista include some additional considerations. For more information, see the “Microsoft Vista installations” section on page 2–2.

- Sufficient disk space for the WebClient and the application files

  **Note:** During installation, you should shut down non-essential programs and processes to free up DLLs and such.

- Appropriate write privileges for installation and upgrades

  The standard model assumes that the WebClient automatically checks for and installs upgrades when it runs. Any end user has sufficient privileges to upgrade their personal WebClient installations. However, an administrator must grant certain privileges to non-administrators to enable them to upgrade an Administrator WebClient installation. If the end users are not granted those privileges, an administrator would need to arrange batch jobs with proper privileges to keep the WebClient application up-to-date. For more information on this, see the “Granting write privileges” section on page 6–3.

- If running the installer over the Web, a Web browser that supports ActiveX controls or plug-in technology, such as Microsoft Internet Explorer 6 and Firefox 2 (or later versions).
WebClient architecture

This section discusses the following aspects of WebClient architecture:

- WebClient Application Assembler
- WebClient Deployment Packager
- Using IntelliStream technology
- Using non-IntelliStream technology
- Using IntelliStream with non-IntelliStream technologies
- Running the application
- Single sign-on
- Server sharing

**WebClient Application Assembler**

The WebClient Application Assembler lets you easily create the files you need to deploy and update WebClient applications. After you complete your application development and compile your application code, you use the WebClient Application Assembler to generate files for deploying and updating the front end of your applications. This tool aids in the following tasks for deploying and updating of your WebClient application:

- Creating and updating a WebClient project (.wcp) file.
- Generating an application configuration (.prowcapp) file for deploying the application, and packaging it into a cabinet (.prowcapc) file.
- Identifying components and generating cabinet (.cab) files that contain your application files and updates to your application files.
- Providing a set of features to support the installation of your application.

For more information, see Chapter 6, “Deploying an Application.”

Using the Application Assembler, you can implement installations and updates that use IntelliStream technology, non-IntelliStream technology, or a combination of the two. For more information, see the “Using IntelliStream technology” section on page 1–12, the “Using non-IntelliStream technology” section on page 1–14, and the “Using IntelliStream with non-IntelliStream technologies” section on page 1–15.
**WebClient Deployment Packager**

The WebClient Deployment Packager is a convenient tool for creating custom deployment configurations after the application configuration is generated by the WebClient Application Assembler. Useful for tailoring the deployment information (such as the location of the server code and information about digital signatures for security) to the needs of specific users, the Deployment Packager eliminates the need to regenerate the `.prowcap` and `.prowcapc` files. A standalone version is provided for distribution to customers or application hosters who want to customize their own deployment configuration. For more information, see Chapter 6, “Deploying an Application.”

**Using IntelliStream technology**

With IntelliStream™ technology, the application deployer can include in the initial download only those application files needed to start the application. Then, as the end user runs other parts of the application, WebClient automatically downloads and installs the additional files as needed.

IntelliStream also works every time the end user starts an application by checking for updated application components, downloading them, and installing them—all before the application starts.

**Note:** Some IntelliStream System Tasks require Administrator privileges to install. If your application makes use of these tasks, users cannot install personal instances of your WebClient application. For more information, see the “System Tasks and personal installations” section on page 6–19.

When an application component reaches the end user’s machine, either as part of the initial download or as part of an update, the files within the component are cached. For more information on IntelliStream, see Chapter 6, “Deploying an Application.”

**Advantages of using IntelliStream**

Using IntelliStream provides the following advantages:

- The initial download and subsequent upgrades do not require special installation procedures or update procedures.
- The initial download does not need to contain all components of the application. Thus, the end user’s machine is not cluttered with application components that are never used.
- When the application deployer updates the application, WebClient on the end user’s machine automatically downloads only the application-component elements that have changed, making downloads faster.
- Application components (for the initial download and for updates) can be hosted on an AppServer as well as on a Web server.

**Codebase locator**

As part of defining the application for WebClient, the application deployer describes the server that contains the application components (for the initial download and for updates) to be downloaded, using the Application Assembler. This description is called the *codebase locator*, which is used by WebClient to access this server, called the *codebase server*. For more information, see Chapter 3, “Hosting the WebClient Deployment.”
**Components and download mode**

To download the files of the client portion of the application, WebClient places them in Microsoft cabinet files. Typically, a single cabinet file might contain several procedure (.r) or procedure library (.p1) files, image files, and so on. A single cabinet file comprises an application component.

The application deployer assigns one of three download modes to each application component:

- **At Startup** — The initial download includes the component.

- **As Needed** — WebClient automatically downloads the component when the end user uses the part of the application that references it.

- **Ask User at Startup** — At the time of the initial download, WebClient prompts end users to identify which As Needed components they want downloaded. WebClient then downloads these components (along with the at-startup components).

For more information, see Chapter 6, “Deploying an Application.”

**Component update files**

When the application deployer defines a new version of an application to the Application Assembler, the Application Assembler automatically creates a component update file for each component whose new version contains changed application files. Each component update file represents the delta (difference) between the new version of a component and the previous version of the same component.

When WebClient on the end user’s machine determines that the end user’s application needs updating, WebClient downloads the necessary component update files to update the end user’s installation to the newest version. For more information, see Chapter 6, “Deploying an Application.”

**Note:** IntelliStream does not support incremental updates to memory-mapped procedure libraries. For more information, see the “Updating procedure libraries” section on page 6–43.

**Determining if a resource was modified**

When the application deployer generates a new version of an application using the Application Assembler, the Application Assembler determines if an r-code resource has changed since the last version by comparing the MD-5 values of each version of the r-code from a version-information file. (MD-5 is an algorithm that reads a file and produces a short mathematical value. If a file changes even slightly, the MD-5 value changes.) For more information on the version-information file, see Chapter 6, “Deploying an Application.”

When compiling procedures, be sure to specify the Generate MD-5 option. If you do not, MD-5 values are not inserted into r-code resources, forcing the Application Assembler to check for changes much more coarsely (by comparing time stamps, which might cause unnecessary downloads). For more information on compiling procedures for version changes, see Chapter 5, “Developing the Application.”
Obsolete versions

When the deployer defines a new version of the application, they can also define certain previous versions as obsolete. For example, if you know that all end users have upgraded beyond a certain version, you might define any earlier version as obsolete. When WebClient generates a new version, it does not produce component update files for obsolete versions. For more information, see Chapter 6, “Deploying an Application.”

System tasks and ABL routine facilities

The IntelliStream System Tasks facility (in the Application Assembler) can perform an array of built-in installation tasks, such as creating desktop shortcuts and registering system files. In addition to the System Tasks facility, WebClient can invoke a ABL routine you supply to perform additional installation tasks. For more information on these facilities, see Chapter 6, “Deploying an Application.”

Note: Some IntelliStream System Tasks require Administrator privileges to install. If your application makes use of these tasks, users cannot install personal instances of your WebClient application. For more information, see the “System Tasks and personal installations” section on page 6–19.

Digital signatures

WebClient lets the application deployer digitally sign each deployed file. When the end user downloads a file, the end user’s WebClient instance validates the digital signature. For complete information on digital signatures in general and how WebClient uses them in particular, see Chapter 4, “Designing Security.”

Using non-IntelliStream technology

The non-IntelliStream initial download (which includes WebClient and the application) can use Web-based or CDROM-based external installer technology such as InstallShield and similar technologies that can install and uninstall applications.

The CDROM option is for customers who have slow communication lines. But the focus of WebClient is to deploy to end users over the Web.

The WebClient distribution includes an installation image for installing WebClient on the end user’s machine.

To prepare the application for the initial download, the application deployer prepares an installation procedure, using InstallShield One-Click Install technology or any other technology executable from a Web browser.

You can customize the WebClient installation so it automatically starts the application installation.

To start the initial download using a Web browser, the end user goes to the application deployer’s Web site and clicks on a button. This initiates the WebClient, and the application is downloaded and installed on the end user’s machine.
Using IntelliStream with non-IntelliStream technologies

For the initial download and for updates, the application deployer can use IntelliStream, an external installer, or a combination of the two.

The application deployer might use IntelliStream for initial downloads and updates:

- To minimize the time required for downloads
- To avoid having to write an InstallShield script, which can be complicated
- To avoid having to buy the InstallShield product

The application deployer might use an external installer for initial downloads and updates if:

- InstallShield is already being used
- Minimizing downloading time is not critical
- Or there are specific installation requirements that IntelliStream cannot handle

The application deployer might combine the two techniques. For example, an initial download might use an external installer, while updates might use IntelliStream, because it minimizes the number of files that need to be downloaded.
Running the application

Once WebClient and the application are installed, the end user starts the application in one of the following ways:

- Going to the application deployer’s Web site and clicking on a button.
- Clicking on a shortcut that was set up by the installation procedure and that resides on the desktop or in a program folder.
- Going to a Web browser and entering the URL of a bootstrap file on the application deployer’s server.

Each time the end user starts the application, WebClient on the end user’s machine compares the local version of the application to the latest version on the application deployer’s Web site. If the versions differ, WebClient on the end user’s machine updates the local version to match the latest version.

Configuration file locator

When the application deployer uses the Application Assembler, the Application Assembler writes the application definition to an application configuration file. The end user downloads the file as part of the initial download, and each time the end user starts the application.

As part of defining the application, the application deployer defines the server from which the end user downloads the application configuration file. WebClient uses this description, called the configuration file locator, to access the application configuration file on the configuration file server. For more information, see Chapter 3, “Hosting the WebClient Deployment.”

Acceptable-to-run versions

When a new version of the application is defined, a previous version that still runs acceptably can be defined as acceptable to run. For example, an acceptable-to-run version might contain a cosmetic bug that the most recent version fixes.

When the end user starts the application, the end user’s WebClient instance downloads the latest application configuration file and checks if any updates are needed. At that point, WebClient might discover that the end user’s version is defined (in the application configuration file) as acceptable to run. When this occurs, the end user is offered the option of downloading the newer version. If the end user so requests, WebClient downloads and applies the updates. For more information on acceptable-to-run versions, see Chapter 6, “Deploying an Application.”

Single sign-on

Progress Software Corporation recommends that each server the application accesses be protected with a user ID and password. To simplify access to servers, multiple servers might be protected by the same user ID and password. To facilitate single sign-on for the end user eliminates the need for the end user to enter their logon information multiple times, WebClient caches authentication information that the end user enters, and gives the application access to this cache.

If the end user so chooses, WebClient maintains the authentication cache across end user sessions. This is called persistent caching. The application deployer can disable persistent caching, and the end user can clear the persistent cache. For more information, see the “Implementing single sign-on” section on page 5–7.
Server sharing

WebClient and the application might share a single server. For example, an AppServer that contains application components downloaded by WebClient might also contain business logic accessed by the same application. When WebClient and the application access the same server, they can share a connection. For more information on server sharing, see the “Managing AppServer connections” section on page 5–6.
Application requirements

As with all modern applications, Progress Software Corporation recommends that you follow modern best practices when designing an application that you plan to deploy with WebClient. The following sections cover some points that are especially important.

Access databases only through an AppServer

A WebClient application must access databases through an AppServer as the client cannot directly access a database. All database data presented by the application to the end user must originate on an AppServer. All database data entered by the end user must be sent to an AppServer for evaluation and storage. To pass relational-database data (such as rows of data) between AppServer and client, you can use temp-table parameters in remote procedures. For more information on temp-table parameters and remote procedures, see *OpenEdge Getting Started: ABL Essentials*.

Modularize code by function

Imagine you have just designed a new application’s procedures and user-defined functions, and you are now assigning each piece of code to a procedure (.p) file. At application-deployment time, you might want only certain components (groups of application files) to download initially, and other components to download only when they are first called. At design time, you can make deploying the application more efficient by the way you assign code to procedure files. Specifically, avoid assigning pieces of code that support dissimilar functions to the same procedure file, since you might want unrelated modules of an application to download at different times.

Note: However, you do not need to place all code that supports similar functions to the same procedure file, since a single component can contain multiple files.

For example, consider an insurance application that supports several different lines of insurance: life, health, automobile, fire, and disability. Assume the application has separate code supporting each line of insurance, plus general-purpose routines used throughout the application. To modularize this application by function, make sure that:

- A single procedure file does not contain routines for more that one line of insurance.
- A procedure file containing general-purpose routines does not contain specific routines for one type of insurance.

Note: If an existing application consists entirely of procedures and user-defined functions that are completely unmodular, with every routine calling every other routine, it might be difficult to have the application download incrementally. However, you can deploy the application as a single large component. For more information, see Chapter 6, “Deploying an Application.”
Choosing which installation types to support

Administrator installations and personal installations have different requirements. You must consider these requirements while designing your application and its WebClient deployment. An Administrator installation requires the end users have write access to certain areas that are normally restricted to administrators, but it imposes no additional restrictions on your designs. A personal installation does not have that write access, which results in some limitations. For example, a personal installation cannot:

- Install files to certain areas. Adding such files through the IntelliStream System Tasks causes install errors for a personal installation. For more information, see the “System Tasks and personal installations” section on page 6–19.

- Install a .NET Framework. This might cause problems if your WebClient application uses the OpenEdge GUI for .NET. For more information, see the “.NET Framework” section on page 6–44.

- Write to registry keys or files located in areas to which the non-administrator does not have access.

**Note:** Your application can check the installation type of a connected client through the SESSION system handle. For more information, see the “Checking application registry access” section on page 5–5.
Designing Your End User’s Experience

Before you begin your application development, you must make decisions about the following:

- How end users install WebClient and WebClient applications.
- How to provide updates for your application.
- How end users access and start your application.
- The type of server to use to host and deploy your application.
- What type of documentation you need to provide.

The decisions you make affect your end users and how you develop and deploy your application. To help you make these design decisions, this chapter provides information about:

- Installation process
- WebClient application installation
- Updates for your application
- Uninstalling your application
- Starting WebClient applications
- Providing documentation
Installation process

You must provide a way for your end users to install WebClient on their machines. You have the following options to provide a WebClient installation:

- Host an installation on a Web server
- Point your users to the WebClient installation page
- Provide your users with a CD-ROM with a WebClient installation program

OpenEdge provides the following WebClient installers:

- A stand-alone InstallShield image that you can use to create an installation CD-ROM.
  
  **Note:** You cannot customize the CD-ROM installation.

- A One-Click Install image that you can use to host a Web-enabled installation.
  
  **Note:** Progress Software Corporation strongly recommends customizing the One-Click Install image to automatically install the application after installing WebClient, and to start the application after it is installed.

Both installers install WebClient on the user’s machine but provide a different experience for your end user:

- With a CD-ROM installation, your end users use a setup program to install WebClient.
- With a Web-enabled installation, your end users go either to a Web page you provide or to the Progress Software Corporation Web site that provides a link for installing WebClient.

**Note:** You can perform a silent install of WebClient for either an Administrator installation or a personal installation. The variables set in a response file are the same.

For more information about WebClient installers provided with OpenEdge and how to customize and prepare a WebClient installer for deployment, see Chapter 6, “Deploying an Application.”

Microsoft Vista installations

Installing WebClient and WebClient applications in Microsoft Vista includes additional considerations due to Vista’s User Access Control (UAC) feature. With UAC enabled, an Administrator user runs applications as a Standard (unprivileged) user. To use their additional privileges, an Administrator must specifically select the **Run As Administrator** option from a context menu. By running the Web browser as a Standard user for the installation process, an Administrator could install personal instances of WebClient and WebClient applications in Vista, which is not possible in previous versions of Windows.

**Note:** Progress Software Corporation recommends that Administrators do not create personal instances for themselves in this manner.
Another Vista feature that impacts installations of applications that have not been adapted to Vista is “virtualization.” This feature causes those applications to redirect writes for protected areas of the registry and file system to a private area for each user. This virtual access to those protected areas enables unadapted applications to still function, while maintaining Vista’s security control over protected areas.

After performing an Administrator installation, a member of the Administrator group must grant the end users write access as described in the “Granting write privileges” section on page 6–3. However, an Administrator must perform any uninstalls using the Run As Administrator option.
WebClient application installation

When you design your application, you must decide on the installer technology you want to use. You can deliver your application on CD-ROM, over the Internet, or over a company intranet. OpenEdge offers great flexibility for installing your application. You can use:

- The WebClient IntelliStream technology to install front-end components for your application and perform optional system tasks to satisfy basic installation requirements. This can include an ABL procedure to complete installation tasks that IntelliStream could not complete.

- An external installer, such as InstallShield Professional, which includes the One-Click Install technology for installing the entire front end of your application over the Web, or technology to create a CD-ROM for your application installation.

**Note:** You do not have to use the same technology to install your WebClient application as you do for installing WebClient.

For more information about using WebClient IntelliStream, see the “IntelliStream” section on page 2–5. For more information about using an external installer, see the “External installer” section on page 2–6.

Progress Software Corporation recommends that you provide a Web-based installation, either IntelliStream or a Web-enabled external installer, as the typical approach. For users without access to high-speed Internet connections, you might want to provide a CD-ROM installation. You should provide all updates over the Web.

When designing your WebClient application delivery method, consider the following recommendations:

- Use IntelliStream technology to install your application over the Internet or an intranet when you want to take advantage of the benefits that it offers. IntelliStream technology can:
  - Initially install only the required front-end components of your application, and then install other components only when a user needs the functionality
  - Optionally install your application from an AppServer with integrated security
  - Automatically start the application after installation
  - Save you from buying and learning another installer technology

- Use an external install product, such as InstallShield® One-Click Install™ technology, to install the application from a Web page when IntelliStream does not support functionality you need to install the application.

- Use an external install product, such as InstallShield, when you want your end users to install the entire application from a CD-ROM.
• If you need to provide both CD-ROM and Web installation for different customers, you can:

  – Use an external install product, such as InstallShield, for CD-ROM installations, and use IntelliStream for installations over the Internet or an intranet. This design provides the benefits of IntelliStream features to those end users who have fast Internet access.

  – Use an external install product, such as InstallShield, for CD-ROM installations, and the One-Click technology for installations over the Internet or an intranet.

**IntelliStream**

You can use IntelliStream to have WebClient install your application automatically with minimal interaction by your end users. IntelliStream includes an installer, built into WebClient, that installs and updates your application by downloading the required application components or component updates.

When you use IntelliStream, WebClient reads the application configuration file to determine which front-end components to download initially. Depending on how you implement your application installation, your users might need to respond to prompts for security information and information about the components they want to download. For more information, see Chapter 4, “Designing Security” and Chapter 6, “Deploying an Application.”

IntelliStream provides the following features:

• Automatic download of component (.CAB) files

• Creation of an **Add/Remove Program** entry for uninstalling the application

Optionally, IntelliStream can:

• Set up application shortcuts on the desktop or in the Program folder

• Run `ini2reg` to store your `.ini` file information in the registry

• Install and register application and system files

• Run a ABL installation procedure that can complete application-specific installation tasks

When you use IntelliStream, you use the WebClient Application Assembler to define and generate the application component (.CAB) files and to choose the optional installation tasks that you want IntelliStream to perform.
External installer

If you decide to use an external installer, such as InstallShield or a similar program, Progress Software Corporation recommends that your installer perform all the installation tasks required to install the entire application. When you use an external installer, the installer should:

- Copy all required files to the end user’s machine
- Create registry entries for WebClient and your application
- Provide an uninstallation program for the application, typically using the same technology as the installation

Optionally, the installer can create a shortcut for running the application and can perform any additional tasks you want the installation to complete.

For more detailed information about installing your application, see Chapter 6, “Deploying an Application.”
Updates for your application

You must choose a technology to update your application. To provide updates, you can use:

- **IntelliStream technology** — When you use IntelliStream, you use WebClient Application Assembler to generate the deployment files necessary to update your application.

  **Note:** You can use IntelliStream to update your application even if you use an external installer to perform the initial installation. Progress Software Corporation recommends IntelliStream for updates, regardless of the technology used to perform the initial installation.

- **The same update method as your external installer technology** — When you use an external installer technology, you must provide a Web-enabled installation for updates.

For more detailed information about updating your application, see Chapter 6, “Deploying an Application.”
Uninstalling your application

Typically, installer technology includes an uninstallation feature. For example:

- If you use IntelliStream to install your application, WebClient provides the uninstallation technology for your application.

- If you use an external installer to install your application, your uninstallation program must be made available to remove the application from the user’s machine.

- If you use an external installer to install your application and IntelliStream to update your application, then the external installer’s uninstallation program must call WebClient Initializer in uninstall mode to remove any update files and registry key values that IntelliStream added.

Since they cannot write to the correct registry keys, users without Administrator privileges cannot uninstall an Administrator installation. Since they cannot see another user’s personal installation, no user can uninstall another user’s personal installation.

If an uninstall fails, you can perform a manual uninstall of WebClient following the same general procedure as outlined for uninstalling OpenEdge in OpenEdge Getting Started: Installation and Configuration.

**Caution:** If you use IntelliStream to install or update your application and you plan to uninstall the WebClient, then you must uninstall your application before you uninstall the WebClient.

For more information, see Chapter 6, “Deploying an Application.”
Starting WebClient applications

You can design your WebClient application deployment so that the application is started by:

- A URL to a bootstrap Web page, so that the user can start the application from a Web browser. You must use this bootstrap process to install and run the application initially, and whenever you change the location of the configuration file.

  **Note:** Your users also can use this bootstrap Web page to initiate the installation of WebClient.

- A shortcut that starts the application directly from the Windows desktop after the application is installed.

For more information, see Chapter 3, “Hosting the WebClient Deployment” and Chapter 6, “Deploying an Application.”
Providing documentation

Your deployment plan should include providing documentation for your end users. You have a number of options that can include printed or electronic documentation. Whatever you decide, the documentation must explain:

- How to install WebClient.
- How to install and uninstall the application.
- How to access and run the application.
- Optionally, how to manage application functions with the WebClient Application Manager that is installed with WebClient. For example, you might need to provide instructions for setting user-specific startup parameters required by your application.

This chapter provides an overview of deployment design considerations you must make for your application. For more detailed information about deploying your WebClient applications, see Chapter 6, “Deploying an Application,” and Chapter 7, “Your End Users’ Experience.”
Hosting the WebClient Deployment

When you deploy a WebClient application, you must choose the servers to host your application. This chapter discusses the following choices:

- Hosting application configuration files
- Hosting codebases
Hosting application configuration files

Whether you use the IntelliStream technology or not, WebClient requires downloading the application configuration file from a standard Internet-based server, either a Web server or a file server.

You might host the application configuration file on a server you have already set up. For example, if you have already set up a Web server with the AppServer Internet Adapter (AIA), you might host the configuration file on this Web server. Or, if you have already set up a file server (and are working entirely within a corporate intranet and are using file sharing), you might host the configuration file on the file server.

After selecting a server, you describe the server to WebClient through the WebClient Application Assembler’s **Configuration File Locator Definition** window at application-deployment time:

When WebClient downloads the application configuration file from its server, the download uses a communication protocol based on the server’s type. If the application configuration file resides on a Web server, downloads can use either HTTP or HTTPS. HTTPS encrypts the download, which increases security. HTTP does not encrypt the download, but is somewhat faster.

If the application configuration file resides on a file server, downloads use the File protocol.

The URL you enter in the Application Assembler’s **Configuration File Locator** window tells WebClient which protocol to use.

**Note:** Progress Software Corporation recommends that for access over the Web, you select a communication protocol that encrypts downloads. For more information, see Chapter 4, “Designing Security.”

For more information on URLs and communication protocols, see *OpenEdge Application Server: Administration.*
Hosting codebases

If you are using IntelliStream, the codebase (the application components for the initial download and for updates) can reside on a standard Internet-based server, or on an AppServer.

Just as with the application configuration file, you might host the codebase on a server you have already set up. For example, if you have already set up an AppServer with the business logic that the application accesses, you might host the codebase there.

After selecting a server, you describe the server to WebClient through the Application Assembler’s Codebase Locator Definition window at application-deployment time:

Hosting the codebase on an internet-based server

Hosting the codebase on an Internet-based server has the advantage of allowing the application configuration file and the codebase to be managed consistently. If the application configuration file resides on a Web server, downloads can use either HTTP or HTTPS. HTTPS encrypts the download, which increases security. HTTP does not encrypt the download, but is somewhat faster.

If the application configuration file resides on a file server, downloads use the File protocol.

Hosting the codebase on an AppServer

Hosting the codebase on an AppServer has the following advantages:

- If the codebase can reside on the same AppServer that hosts the business logic, administrative tasks are simplified.

- If the AppServer hosting the business logic already has security (such as user IDs and passwords), placing the codebase on that AppServer protects the codebase with the same security, at little or no additional cost.

But, hosting the codebase on an AppServer has the disadvantage that it adds additional load to the AppServer. You might need to configure an additional AppServer instance to handle the additional load of hosting the codebase and to avoid performance degradation.
If application components reside on an AppServer, downloads can use the HTTP, HTTPS, or any of the four variations of AppServer protocol. The URL you enter in the Application Assembler’s Codebase Locator window tells WebClient which protocol to use for codebase downloads. To choose among these protocols, consider the following:

- Using the HTTP or HTTPS protocol with an AppServer requires using the AIA.
- The HTTPS protocol encrypts downloads over the Internet, which increases security.
- The HTTP protocol does not encrypt downloads, but is faster than HTTPS.
- The AppServer protocols do not use the AIA.
- The AppServerS and AppServerDCS protocols use SSL tunneling to encrypt downloads over an intranet, which increases security.
- The AppServer and AppServerDC protocols do not encrypt downloads, but are faster than their SSL-based counterparts.

For more information on URLs and communication protocols, see *OpenEdge Application Server: Developing AppServer Applications*. 
Designing Security

Because of the ever-present security threats when using the Web to distribute your applications, you must secure the files you distribute. Ideally, your application should combine tight security with a clean user interface that does not require end users to enter their logon information multiple times. For more information on security features in OpenEdge, see the chapter on security in *OpenEdge Getting Started: Core Business Services*.

There are a number of ways to provide security, including digitally signing files, using user IDs and passwords, caching information, and using secure communication protocols. Information about these different methods is contained in the following sections:

- Digitally signing files
- Protecting servers with user IDs and passwords
- Single sign-on and security caching
- Choosing secure communication protocols
Digitally signing files

When deploying an application, you can digitally sign each cabinet file and have the end user verify each digital signature when the file downloads. This approach provides end users with the assurance that each file has:

- **Authenticity** — Assurance that you prepared the file
- **Integrity** — Assurance that nobody has tampered with the file

**Note:** Progress Software Corporation strongly encourages the application deployer to digitally sign each file to be downloaded.

How WebClient uses digital signatures

If you want to digitally sign each cabinet file for your WebClient application and you want the end user to verify the digital signature of the downloaded files, who needs which key?

To digitally sign a cabinet file, you need:

- Your private key
- Your public key in the form of a public-key certificate

To verify the digital signature of a downloaded cabinet file, the end user needs your public key in the form of a public-key certificate.

So, to use digital signatures, you need a private key, a public key, and a public-key certificate, while your end user needs your public-key certificate. This section covers the following topics:

- Getting a private key, public key, and public-key certificate
- Defining an application as signed
- How your public-key certificate gets to the end user
- Creating test public-key certificates

Getting a private key, public key, and public-key certificate

The following section describes the process for obtaining private and public keys.

To get a private key, a public key, and a public-key certificate:

1. Select a PKI vendor (CA) whose software is compatible with Microsoft Authenticode Technology and request a “software publishing digital certificate.”

   To get names of CAs, ask your PSC Product Marketing representative.

2. Install the software that generates and securely stores public keys and private keys on your system.

   You can typically get the software from Microsoft or you can download it from the CA’s Web site. You might have to provide a name for the certificate storage location.
3. Fill out the CA’s request for information about you, your company, and how you are going to pay.

4. Submit the requested information and the stored public key to the CA.

5. Wait for the CA to verify your identity.

   **Note:** The CA might use phone calls or personal visits to verify the information you supply.

6. If the CA can prove your individual and corporate identity, they will contact you and tell you how to obtain your digital certificate. Typically, this involves the same software and the same Web site that you used to apply.

   The digital certificates are stored on your system in the same named certificate location as the one used for the initial public/private key generation.

   **Note:** You can have digital certificates issued by multiple CAs for a single public/private key pair.

---

**Defining an application as signed**

Now that you have your keys and certificate, you can define an application as signed. To do so, go to the WebClient Application Assembler’s **Generate** window and click the **Security** button to launch the **Security** window. In the **Digital signature for generated files** group box, check **From registry** (if the digital signature information resides in the registry) or **From file** (if the digital signature information resides in a file):

![Security window](image)

When you define an application as signed and you generate the application, the Application Assembler adds a copy of your public-key certificate to each cabinet file and digitally signs them.
How your public-key certificate gets to the end user

When an application is defined as signed and the end user downloads a signed configuration or component cabinet file (each of which contains your public-key certificate), WebClient on the end user’s machine:

1. Extracts the digital signature and your public-key certificate from the cabinet file.
2. Verifies the digital signature of the cabinet file, using your public-key certificate.
3. Also verifies your public-key certificate through its issuer’s root public-key certificate. The issuer’s root public-key certificate can be obtained from the cabinet file itself or from the certificate store used by Microsoft Internet Explorer.
4. Displays the information on the certificate and asks whether the end user trusts it.

**Note:** If the end user says no, the process aborts.

5. Optionally stores your public-key certificate in the digital certificate store of Internet Explorer.

Creating test public-key certificates

WebClient includes a batch file, MakeTestCert.bat, that makes it easier for you to create mock public-key certificates for testing. For more information, see the comments in the file, which resides at OpenEdge-Install-Directory\bin.
Protecting servers with user IDs and passwords

You can require user IDs and passwords for each Internet-based server and AppServer accessed by WebClient or the WebClient application. Although user IDs and passwords are not required, using them strengthens security.

**Note:** Progress Software Corporation recommends that you seriously consider protecting each server accessed by WebClient or the application with a user ID and password.

For more information on user IDs and passwords for a particular Internet-based server, see the documentation for that server. For more information on setting user IDs and passwords for an AppServer, see *OpenEdge Application Server: Developing AppServer Applications*.

Prompting the end user for authentication information

Instead of embedding authentication information in a server URL, you can have WebClient prompt the end user for them when they are needed. How you enable the prompting depends on whether you are configuring an application-configuration-file server or a codebase server.

Enabling authentication for an application-configuration-file server

To enable authentication for an application-configuration-file server, go to the Application Assembler’s *Configuration File Locator Definition* window. In the *Authentication Information* group box, fill in the *End-User Description* field and check the *Prompt for ID/Password (Unless Cached)* box:

![Configuration File Locator Definition](image)

**Note:** WebClient uses this information only if the end user starts an already-installed WebClient application by using a shortcut. For more information on starting an already-installed application with a shortcut, see Chapter 6, “Deploying an Application.”
Enabling this option displays the following authentication prompt to the end user:

![Authentication Information Window]

### Enabling authentication for a codebase server

To enable authentication for a codebase server, go to the Application Assembler’s **Codebase Locator Definition** window. In the **Authentication Information** group box, fill in the **End-User Description** field. Then, check the **Prompt for ID/Password (Unless Cached)** box, and finally, if the codebase server is an AppServer, check the **Prompt for AppServerID/Password/Info String (Unless Cached)** box:

![Codebase Locator Definition Window]

Enabling this option displays the following authentication prompt to the end user:

![Authentication Information Window]
Single sign-on and security caching

WebClient caches each user ID and password entered by the end user and retrieves cached authentication information to access additional objects that require the same authentication. When the end user provides authentication information to connect to a server and to download files, WebClient can cache the authentication information and make it available to the application. Similarly, when the end user provides authentication information at an application prompt to connect to a server that contains business logic, the application can make the authentication information available to WebClient. By default, WebClient maintains a separate security cache for the application-configuration-file server and the codebase server.

Security caching lets you implement single sign-on, which keeps end users from being prompted multiple times for the same authentication information. Single sign-on is useful when:

- Connecting to the same server multiple times during one session or across multiple sessions; for instance, when the same AppServer handles both downloads of application components and running business logic for the application. For example, if WebClient downloads application components before the application accesses the business logic, the application can automatically connect to the AppServer to access the business logic using the same authentication information. Similarly, if the application accesses the business logic before WebClient downloads application components, the application can make authentication information available to WebClient when it connects to the AppServer to download application components.

- Connecting to multiple servers that use the same authentication information. Suppose the application configuration file and the application components are on different servers, but you set the same authentication information for both servers. You can direct WebClient to make the configuration file’s authentication information available to the application as codebase authorization information (but not the other way around).

Note: Authentication information is always stored encrypted.

Sharing the configuration file cache with the codebase cache works only if the application is launched from a shortcut, not from a Web browser—because in the latter case, the configuration file is downloaded by the Web browser, whose cache WebClient cannot access.

By default, WebClient does not maintain security caches on a particular machine across sessions. To override this default behavior, the end user must specifically request the persistent cache.

You can tell WebClient to disable the persistent cache. If you do so, the end user does not have the option of saving authentication information across sessions, and WebClient deletes the security caches at the end of each WebClient session.

Note: If persistent caching is not disabled, an end user can choose to persistently cache their authentication information for particular servers. After that, anyone starting a new WebClient session at the same machine and logging in as the original end user can access those servers without having to re-enter the authentication information.
Using WebClient logon dialogs in your applications

Suppose your application uses the same AppServer for application components downloads and for accessing business logic, and suppose further that an application needs to access the AppServer before WebClient downloads any As Needed application components from the AppServer. (An application can determine this because the security cache does not contain authentication information such as user IDs or passwords.) As a result, the application must prompt the end user for the logon information. To do so, your application can display the security dialogs that WebClient uses, providing the end user with a consistent interface.

For information on calling the authentication dialogs that WebClient uses from the application, see the “Using WebClient’s authentication dialogs” section on page 5–8.
Choosing secure communication protocols

To guard against the inherent security problems associated with network traffic, and to provide a greater level of security for your applications, you can encrypt an application’s configuration file and application-component downloads by using a secure communication protocol. For example, you can use HTTPS to encrypt network traffic with a combination of symmetric and asymmetric ciphers. For more information on symmetric and asymmetric ciphers, see the “Digitally signing files” section on page 4–2.

Secure protocols that you can use, depending on the server type (Web server, file server, or AppServer) where the files are hosted, include:

- HTTPS (requires the use of an AIA)
- AppServerS (codebase downloads only)
- AppServerDCS (codebase downloads only)

For more information on communication protocols for application configuration file and codebase downloads, see Chapter 3, “Hosting the WebClient Deployment.” For more information on the HTTPS protocol and the AIA, see OpenEdge Application Server: Administration.
Developing the Application

Although coding an application for WebClient closely resembles coding an application for the ABL client, WebClient provides several additional features you can take advantage of, as outlined in the following sections:

- Using the CODEBASE-LOCATOR handle and its attributes
- Checking application registry access
- Managing AppServer connections
- Implementing single sign-on
- Using WebClient’s authentication dialogs
- Using URLs in PROPATH for image file downloads
- Compiling the application
Using the CODEBASE-LOCATOR handle and its attributes

Once the application configuration file contains the codebase server description, WebClient makes the description available, to the client at run time, as attributes of the CODEBASE-LOCATOR handle. This section discusses these attributes, which are organized in logical groups.

**Note:** All the attributes discussed in this section are read-only.

### Basic attributes

Table 5–1 lists the basic attributes of the CODEBASE-LOCATOR handle.

<table>
<thead>
<tr>
<th>Name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>END-USER-PROMPT</td>
<td>Character</td>
<td>Text string for the security prompt of an authentication dialog. Used only if WebClient must prompt for security information.</td>
</tr>
<tr>
<td>KEEP-CONNECTION-OPEN</td>
<td>Logical</td>
<td>If TRUE, any connection WebClient creates to download As Needed components will be kept open until all As Needed components have been downloaded to the end user's machine.</td>
</tr>
<tr>
<td>LOCATOR-TYPE</td>
<td>Character</td>
<td>&quot;AppServer&quot; or &quot;InternetServer.&quot;</td>
</tr>
<tr>
<td>NEEDS-APPSERVER-PROMPT</td>
<td>Logical</td>
<td>If TRUE, enables prompting for a user ID, password, and info string for accessing an AppServer. Applies only if LOCATOR-TYPE is &quot;AppServer.&quot;</td>
</tr>
<tr>
<td>NEEDS-PROMPT</td>
<td>Logical</td>
<td>If TRUE, enables prompting for a user ID and password for accessing the codebase server. If LOCATOR-TYPE is &quot;InternetServer,&quot; the prompt is for accessing a codebase residing on an Internet-based server. If LOCATOR-TYPE is &quot;AppServer,&quot; the prompt is for accessing the AIA for a codebase residing on an AppServer.</td>
</tr>
<tr>
<td>PERSISTENT-CACHE-DISABLED</td>
<td>Logical</td>
<td>If TRUE, the user is not given the option of saving authentication information past the end of the session.</td>
</tr>
<tr>
<td>URL</td>
<td>Character</td>
<td>URL of the server.</td>
</tr>
</tbody>
</table>
Security-cache attributes

Table 5–2 lists the attributes of the CODEBASE-LOCATOR handle that relate to the security cache.

Table 5–2: Security-cache attributes of the CODEBASE-LOCATOR handle

<table>
<thead>
<tr>
<th>Name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>APPSERVER-INFO</td>
<td>Character</td>
<td>Info string to use for connecting to the AppServer. Unused if LOCATOR-TYPE is “InternetServer.”</td>
</tr>
<tr>
<td>APPSERVER-PASSWORD</td>
<td>Character</td>
<td>Password for connecting to the AppServer. Unused if LOCATOR-TYPE is “InternetServer.”</td>
</tr>
<tr>
<td>APPSERVER-USERID</td>
<td>Character</td>
<td>User ID for connecting to the AppServer. Unused if LOCATOR-TYPE is “InternetServer.”</td>
</tr>
<tr>
<td>KEEP-SECURITY-CACHE</td>
<td>Logical</td>
<td>If TRUE, WebClient saves the security-cache attributes and this attribute, restoring them the next time the end user reruns the application. The default is FALSE. Set to TRUE if the end user sets the Save Authentication Information for Future Use toggle box. Can also be set to TRUE by the application, depending on the security dialog used. The application might set this before WebClient does. For more information, see the “Implementing single sign-on” section on page 5–7. <strong>Note:</strong> The PERSISTENT-CACHE-DISABLED attribute must be FALSE to use this attribute.</td>
</tr>
<tr>
<td>URL-PASSWORD</td>
<td>Character</td>
<td>Password for accessing the server. If LOCATOR-TYPE is “InternetServer,” the password is for accessing a codebase on an Internet-based server. If LOCATOR-TYPE is “AppServer,” the password is for accessing AIA for a codebase on an AppServer. Unused if the URL does not take a password (for example, for a URL of type FILE).</td>
</tr>
<tr>
<td>URL-USERID</td>
<td>Character</td>
<td>User ID for accessing the server. If LOCATOR-TYPE is “InternetServer,” the user ID is for accessing a codebase on an Internet-based server. If LOCATOR-TYPE is “AppServer,” the user ID is for accessing AIA for a codebase on an AppServer. Unused if the URL does not take a user ID (for example, for a URL of type FILE).</td>
</tr>
</tbody>
</table>
Additional attribute

Table 5–3 lists an additional attribute of the CODEBASE-LOCATOR handle.

**Table 5–3: Additional attribute of the CODEBASE-LOCATOR handle**

<table>
<thead>
<tr>
<th>Name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
</table>
| SERVER| Handle    | Handle to the AppServer that WebClient should use to perform the next application-component download.  
|       |           | Initialized to the Unknown value (?).                                                         |

At startup, WebClient initializes SERVER to the Unknown value (?).

The main purpose of this is to use the same AppServer connection for accessing business logic as for downloading as-needed components. To do this, the application sets the SERVER attribute once the AppServer connection is established.

If SERVER is not set by the application, then, after WebClient downloads the first as-needed component, WebClient sets SERVER to the value of the AppServer’s handle if all of the following are true:

- The server’s TYPE is “AppServer”
- KEEP-CONNECTION-OPEN is TRUE
- At least one as-needed component remains to be downloaded
Checking application registry access

If users can install your application as either Administrator or personal instances, your application might need logic to alter where it reads and writes registry information based on the type of installation. To enable this, the SESSION system handle has a new attribute, WC-ADMIN-APP. The following code snippet is an example of such logic:

```
DEFINE VARIABLE cAppKey AS CHARACTER NO-UNDO.
IF SESSION:WC-ADMIN-APP THEN
  cAppKey = "HKEY_LOCAL_MACHINE".
ELSE
  cAppKey = "HKEY_CURRENT_USER".
LOAD "MyEnvironment" BASE-KEY cAppKey.
```

For more information on this attribute, see OpenEdge Development: ABL Reference.
Managing AppServer connections

To download as-needed application components from a codebase residing on an AppServer, WebClient, by default, creates a new connection to the AppServer, downloads the components, and terminates the connection.

**Note:** To download at-startup application components before the application starts, WebClient creates a new connection to the codebase server.

If you want, WebClient can keep the connection used for downloading as-needed application components open and reuse the connection to download additional components until no more remain to be downloaded. To enable this option, in the Application Assembler’s **Codebase Locator** window, check the **Keep Connection Open** toggle box.

WebClient can also download as-needed application components over an AppServer connection that is already open. For example, if your business logic and the codebase reside on the same AppServer, you could use a connection that is already established to access the business logic. To enable this option, just after you connect to the AppServer, assign the connection’s handle to the **SERVER** attribute of the **CODEBASE-LOCATOR** handle.

If you do so:

- The application has full responsibility for managing the connection.
- The application configuration file’s information regarding the codebase server is not used.
- If the connection breaks, the download fails.
- If there is an outstanding **ASYNC** request on the handle of the existing connection at the moment WebClient needs it for the download, WebClient raises an error on the **RUN** statement.

**Note:** For this reason, Progress Software Corporation recommends that an AppServer connection used for **ASYNC** requests not be used for downloading application components.
Implementing single sign-on

The application can use attributes of the CODEBASE-LOCATOR handle to implement single sign-on, which eliminates the need for the end user to enter the same user ID or password multiple times. Single sign-on is illustrated by the following scenarios:

- WebClient prompts the end user for authentication information and stores the end user’s responses in the security cache. Then, the application accesses the cached authentication information, using the CODEBASE-LOCATOR attributes to connect to the AppServer if both the codebase and the business logic reside on the same server.

- The application deployer checks the Share Authentication Cache of Configuration File Locator toggle box on the Application Assembler’s Codebase Locator Definition dialog. After the end user enters values for URL-USERID and URL-PASSWORD to download the configuration file, WebClient makes these values (or those that already reside in the persistent cache) available through the CODEBASE-LOCATOR attributes. These can be used to connect to the AppServer through AIA if the same Web server hosts AIA and the configuration file.

If the application requests authentication information before WebClient does, the application might set some or all of the security-cache attributes, which WebClient could then pick up. This might happen if WebClient starts the application, finds that nothing has changed since the last time it ran, and so does not download any components or updates. Then, the application starts, connects to a server, discovers it is the first to connect, and populates these attributes.

If the application sets any of these attributes, the next time WebClient needs to connect to a server, WebClient uses these cached values, rather than prompting the end user again. For more information on these attributes, see the “Using the CODEBASE-LOCATOR handle and its attributes” section on page 5–2.
Developing the Application

Using WebClient’s authentication dialogs

Suppose your application accesses an AppServer and the following are true:

- The same AppServer contains application components for download and business logic for your application.
- The application detects that the security cache does not contain authentication information.

**Note:** These conditions occur when WebClient has not yet downloaded any as-needed application components from the AppServer.

To access the AppServer in this situation, your application can prompt the end user for security items by displaying the security dialogs that WebClient uses. This provides the end user with a consistent interface. These routines reside, as compiled (.r) files, in the WebClient installation at `OpenEdge-Install-Directory\wcadd`. If the CODEBASE-LOCATOR handle contains only some of the required values, these values can be passed into these routines as default values, allowing the user to fill in the remaining values.

When using the authentication dialogs, consider the following:

- To make the dialogs look exactly the same as when WebClient invokes them, obtain the `prompt` input parameter from `CODEBASE-LOCATOR:END-USER-PROMPT`.
- Before calling an authentication dialog, the application is responsible for obtaining the relevant attribute values from the CODEBASE-LOCATOR handle to pass to these routines. For example, the value of `NOT PERSISTENT-CACHE-DISABLED` should be passed for the second parameter, `enable-persistent-checkbox`.
- After calling an authentication dialog, the application is responsible for setting the appropriate CODEBASE-LOCATOR attributes with the information obtained from the end user.

Displaying the AIA authentication dialog

To display WebClient’s AIA authentication dialog, use the `_getAIAAuthentication.p` procedure, which has the following signature:

```plaintext
DEFINE INPUT PARAMETER prompt AS CHARACTER.
DEFINE INPUT PARAMETER enable-persistent-checkbox AS LOGICAL.
DEFINE INPUT-OUTPUT PARAMETER url-userid AS CHARACTER.
DEFINE INPUT-OUTPUT PARAMETER url-password AS CHARACTER.
DEFINE INPUT-OUTPUT PARAMETER persistent AS LOGICAL.
DEFINE OUTPUT PARAMETER choseOK AS LOGICAL.
```

Use this procedure if the codebase and business logic share authentication information and there is no authentication at the AppServer itself.
To invoke the _getAIAAuthentication.p procedure, call it with a partially qualified pathname that starts with the wcadd directory, as in the following code fragment:

```
RUN wcadd/_getAIAAuthentication.p (parameter-list).
```

**Displaying the AppServer authentication dialog**

To display WebClient’s AppServer authentication dialog, use the _getAppServerAuthentication.p procedure, which has the following signature:

```
define input parameter prompt as character.
define input parameter enable-persistent-checkbox as logical.
define input-output parameter appserver_userid as character.
define input-output parameter appserver_password as character.
define input-output parameter appserver_info as character.
define input-output parameter persistent as logical.
define output parameter choseOK as logical.
```

Use the _getAppServerAuthentication.p procedure if the codebase and business logic reside on the same AppServer, or if they share the same authorization information and authentication is required:

- Only at the AppServer and not at the AIA Web server (or you are not using AIA)

  Or:

- At AIA and at the AppServer, but AIA authorization information is set in the security cache and thus available in CODEBASE-LOCATOR:URL-USERID and CODEBASE-LOCATOR:URL-PASSWORD

  **Note:** In this case, you might instead call _getAIAandAppServerAuthentication.p, passing in the known values and allowing the end user to fill in the missing values.

To invoke the _getAppServerAuthentication.p procedure, call it with a partially qualified pathname that starts with the wcadd directory, as in the following code fragment:

```
RUN wcadd/_getAppServerAuthentication.p (parameter-list).
```
Displaying the AIA and AppServer authentication dialog

To display WebClient's AIA and AppServer authentication dialog, use the 
_getAIAandAppServerAuthentication.p procedure, which has the following signature:

| DEFINE INPUT PARAMETER prompt AS CHARACTER. |
| DEFINE INPUT PARAMETER enable-persistent-checkbox AS LOGICAL. |
| DEFINE INPUT-OUTPUT PARAMETER url-userid AS CHARACTER. |
| DEFINE INPUT-OUTPUT PARAMETER url-password AS CHARACTER. |
| DEFINE INPUT-OUTPUT PARAMETER appserver_userid AS CHARACTER. |
| DEFINE INPUT-OUTPUT PARAMETER appserver_password AS CHARACTER. |
| DEFINE INPUT-OUTPUT PARAMETER appserver_info AS CHARACTER. |
| DEFINE INPUT-OUTPUT PARAMETER persistent AS LOGICAL. |
| DEFINE OUTPUT PARAMETER choseOK AS LOGICAL. |

Use the _getAIAandAppServerAuthentication.p procedure if the codebase and the business logic share authorization information, you are using the AIA, and authorization is required by the AIA server and by the AppServer.

To invoke the _getAIAandAppServerAuthentication.p procedure, call it with a partially qualified pathname that starts with the wcadd directory, as in the following code fragment:

RUN wcadd/_getAIAandAppServerAuthentication.p (parameter-list).
Using URLs in PROPATH for image file downloads

There can be URLs in the PROPATH environment variable of a WebClient client. If there are, and the client executes the SEARCH function or the LOAD–IMAGE( ) method, the URL pathnames are processed along with the other pathnames included in the PROPATH. This feature lets the application access files, especially image files, over the Web.

Using the SEARCH function

This is the syntax of the SEARCH function:

Syntax

```
SEARCH ( opsyst-file )
```

The SEARCH function searches the directories and libraries (including URL pathnames) appearing in the PROPATH environment variable for opsyst-file. The SEARCH function returns the full pathname of the file unless it resides in the current working directory. If the SEARCH function does not find the file for any reason, it returns the Unknown value (?).

If the file is found in a directory specified by a URL, SEARCH returns the full URL pathname of the file, which consists of the URL’s PROPATH entry with the filename appended.

If opsyst-file is a fully qualified URL or a fully qualified pathname, SEARCH checks for the existence directly, and does not search the directories and URLs in the PROPATH.

Note: SEARCH does not download any files.

Using the LOAD–IMAGE( ) method

This is the syntax of the LOAD–IMAGE( ) method:

Syntax

```
LOAD–IMAGE ( filename
[   ,
   x-offset ,
   y-offset ,
   width ,
   height
   ]
 )
```

The LOAD–IMAGE( ) method applies to images and buttons. LOAD–IMAGE( ) reads the image contained in filename. LOAD–IMAGE( ) recognizes PROPATH entries consisting of URLs. If LOAD–IMAGE( ) finds an image in the directory specified by a URL, LOAD–IMAGE( ) downloads the image from the Web server and loads it into local memory directly, bypassing the end user’s disk.

If the filename argument is fully qualified (whether URL or local pathname), LOAD–IMAGE( ) loads the image file directly, without searching directories or URLs in PROPATH.
If `LOAD-IMAGE()` cannot load the specified image file, it returns an error indicating the reason for the failure.

---

**Note:** You can also download images over the Web by using the `LOAD-IMAGE-DOWN()`, `LOAD-IMAGE-UP()`, `LOAD-IMAGE-INSENSITIVE()`, and `LOAD-ICON()` methods. This feature is less useful, however, because these methods are generally called for transient graphical-interface events requiring an immediate response, which a Web download might not be able to satisfy.

---

### Handling errors

If the application uses URLs in the `PROPATH`, the application is responsible for handling all errors involving URLs, connections, and authentication.

#### URL specification errors

If the `PROPATH` contains an ill-formed URL (for example, one with the percent symbol (%), which is not allowed), the following processing occurs:

- When the URL is **not** a fully qualified path, `SEARCH` and `LOAD-IMAGE()` do not raise an error, skip the URL, and continue searching with the next entry in the `PROPATH`.
- When the URL is a fully qualified path, `SEARCH` and `LOAD-IMAGE()` raise an error.

#### Connection errors

If the `PROPATH` contains a correctly formed URL but the connection to the Web server fails, the following processing occurs:

- When the URL is **not** a fully-qualified path, `SEARCH` and `LOAD-IMAGE()` do not raise an error, and searching continues with the next entry in the `PROPATH`.
- When the URL is a fully-qualified path, `SEARCH` and `LOAD-IMAGE()` raise an error.
Authentication errors

If an application uses `SEARCH` or `LOAD-IMAGE( )`, the PROPATH contains a URL, and authentication on a Web server fails, the following processing occurs:

- When the URL is not a fully qualified path, `SEARCH` and `LOAD-IMAGE( )` do not raise an error and searching continues with the next entry in the PROPATH.

- When the URL is a fully qualified path:
  - The `SEARCH` function does not raise an error but returns the Unknown value (?)..
  - The `LOAD-IMAGE( )`, `LOAD-IMAGE-DOWN( )`, `LOAD-IMAGE-UP( )`, `LOAD-IMAGE-INSENSITIVE( )`, and `LOAD-ICON( )` methods all raise error 9368, where `server-name` is the host name of the Web server, as shown:

  ```plaintext
  Unable to download file from the web. Authentication failed for server server-name.
  ```

  You can trap this error, request a user ID and password, and try to download the image again, as in the following example:

  ```plaintext
  DEFINE BUTTON Button1.
  Button1:LOAD-IMAGE
    ("http://userid:password@www.progress.com/myimage.jpg") NO-ERROR.
  IF ERROR-STATUS:GET-NUMBER(1) = 9368 THEN DO:
    /* Display dialog. */
    /* Update URL string. */
    /* Retry load-image. */
  END.
  ```
Compiling the application

When you compile ABL procedures for your WebClient application, you can optimize application deployment performance by specifying Yes for the Generate MD-5 option of the Application Compiler (equivalent to using the GENERATE-MD5 option on the COMPILE statement). Progress Software Corporation strongly recommends that you use this option. When specified, this option generates an MD5 message digest, a fixed length value that is unique for the exact contents of a given source code file. If the source code of the procedure changes, compiling the file with this option generates a different message digest value. The compiler then inserts this message digest value in the generated r-code.

When you later deploy a new version of your application using the WebClient Application Assembler, the Assembler uses a series of criteria to determine if a given procedure file requires an update at existing client sites, as follows:

1. If the old and new versions of the procedure file both contain an MD5 message digest, the Assembler compares the two MD5 values. If they are different, the Assembler marks the new file as an update.

2. If neither file contains MD5 values, the Assembler compares the date and size information for each file. If they are different, the Assembler marks the new file as an update.

3. If one file contains an MD5 value and the other contains only date and size information, the two files cannot be otherwise compared. So, the Assembler assumes that the new file has changed and marks it as an update of the old file.

The MD5 value is the first, fastest, and most precise mechanism that the Application Assembler can use to compare the old and new procedure files of an application. Using this value thus enhances the performance of application assembly and deployment.

For more information on message digests, see OpenEdge Getting Started: Core Business Services. For more information on deploying an application with the WebClient Application Assembler, see Chapter 6, “Deploying an Application.”
Deploying an Application

This chapter discusses how to deploy a WebClient application. It introduces the PRO*Tools that you use for a WebClient deployment: the WebClient Application Assembler and the WebClient Deployment Packager. With the Application Assembler, you describe the deployment configuration and generate the deployment files. The Deployment Packager enables you to customize a deployment after generating the deployment files.

This chapter explains how to use these PRO*Tools. It also discusses considerations for deploying application files on servers, for installing WebClient, and for installing and updating your application. It contains the following sections:

- Preparing to use the WebClient Application Assembler
- Installation type considerations
- Choosing an installation method
- Defining applications in WebClient Application Assembler
- Files generated by the WebClient Application Assembler
- Customizing deployment configurations
- Hosting the application on a server
- Installing WebClient
- Bootstrapping a WebClient application
- WebClient application updates
- WebClient and OpenEdge GUI for .NET
Preparing to use the WebClient Application Assembler

Before you run the WebClient Application Assembler, you should read the design chapters in this manual and decide how you want to deliver and install the application on your end users’ machines. These decisions determine the information that you provide in the various WebClient Application Assembler windows.

Required input files

When you run the WebClient Application Assembler, you must have a root directory for your application, and it must contain:

- All files that the application comprises (.r files, .pl files, image files, and so on).
- All files that require special install processing (System Tasks) for your application. This includes third-party files such as ActiveX controls, system .dll files, image files for shortcuts, and an initialization file.

In addition to the above items, the following must be true:

- All files placed in the components must reside under the Application Root Directory that you specify in the General tab of the WebClient Application Assembler.

Note: If your application uses the OpenEdge GUI for .NET, all assemblies used by the application must be in the assemblies directory under the application root directory. For more information on deploying OpenEdge GUI for .NET applications, see OpenEdge Deployment: Managing ABL Applications.

- The application’s directory structure under the Application Root Directory on the development machine must be the same as the directory structure you want to use on the end users’ machine under the directory where the application is installed.

Note: The WebClient Installation does not include any code from the OpenEdge Application Development Environment (ADE), such as, the adecomm code, the ADM code, or any of the ActiveX controls included with the OpenEdge installation: CSComboBox, CSSpin1, and PSTimer. If your application uses any of this code, you must package it into your application installation with the rest of your application code. For more information about adding the ActiveX controls, see the “Options tab” section on page 6–17.
Installation type considerations

The “WebClient installation types” section on page 1–6 described how an end user can install WebClient either as a Administrator instance or a personal instance. Your WebClient deployment might support only Administrator installs, or it might support both Administrator and personal installs. (Because the installer determines what type of installation to perform based on the person’s privileges, the installer always creates an Administrator instance for an end user with Administrator privileges.) Each installation type that you support has requirements for you to consider.

Note: Progress Software Corporation recommends that you do not have both installation types of the same WebClient application on a single machine. If several users on the same machine use a WebClient application, a single Administrator installation uses less resources.

Administrator installations

To start the install process, the WebClient installer first checks if the user can perform an Administrator install. The installer checks if either of the following conditions is true:

- The user is a member of a group that has Administrator rights.
- The user has write-access to the HKEY_LOCAL_MACHINE\SOFTWARE registry key, the C:\Program Files directory, and the C:\Windows\system32 directory.

Granting write privileges

For security reasons, many companies do not want to grant everyone Administrator privileges. To use an Administrator instance, an Administrator should grant all users of your WebClient application write access to certain registry keys and directories. Granting these privileges might be a problem in some deployment environments. Table 6–1 lists the necessary write privileges for installing and using an Administrator installation.

Table 6–1: Administrator installation required permissions (1 of 2)

<table>
<thead>
<tr>
<th>Task</th>
<th>Requires write access to . . .</th>
</tr>
</thead>
<tbody>
<tr>
<td>Install an application¹</td>
<td>HKLM\SOFTWARE to create a &lt;vendor&gt;&lt;app&gt; subkey</td>
</tr>
<tr>
<td></td>
<td>C:\Program Files\Progress Software\WebClient²</td>
</tr>
<tr>
<td></td>
<td>C:\Program Files\WebClientApps</td>
</tr>
<tr>
<td>Run the application</td>
<td>HKLM\SOFTWARE&lt;vendor&gt;&lt;app&gt; and its subkeys</td>
</tr>
<tr>
<td></td>
<td>C:\Program Files\WebClientApps&lt;vendor&gt;&lt;app&gt;³</td>
</tr>
<tr>
<td>Download an “As needed” component</td>
<td>HKLM\SOFTWARE&lt;vendor&gt;&lt;app&gt;</td>
</tr>
<tr>
<td></td>
<td>C:\Program Files\WebClientApps&lt;vendor&gt;&lt;app&gt; and its subdirectories</td>
</tr>
</tbody>
</table>
An Administrator might avoid granting some of these privileges by setting up batch jobs to handle updates for the users. The Administrator must also arrange the download of all “As needed” components before the end users try to access them.

### Personal installations

If the end-users’ company cannot support Administrator instances, then each end user must install their own personal instance. When the WebClient installer determines that an end user cannot perform an Administrator install, the installer automatically switches to performing a personal install.

Table 6–2 lists the default values that the installer supplies during a personal installation. If the user enters a different value, the installer verifies that the user has write privileges for the new choice.

### Table 6–2: Personal installation default values

<table>
<thead>
<tr>
<th>Directory</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>WebClient install</td>
<td>%USERPROFILE%\Application Data\Progress Software\WebClient</td>
</tr>
<tr>
<td>Program shortcuts</td>
<td>C:\Documents and Settings&lt;user&gt;\Start Menu\Programs\PSC WebClient &lt;release-number&gt;</td>
</tr>
<tr>
<td>WebClient application install</td>
<td>%APPDATA%\WebClientApps</td>
</tr>
<tr>
<td><strong>Note:</strong></td>
<td>In some circumstances, the installer might choose a different default:</td>
</tr>
<tr>
<td></td>
<td>• %LOCALAPPDATA%\WebClientApps</td>
</tr>
<tr>
<td></td>
<td>(Vista only, non-roaming data)</td>
</tr>
<tr>
<td></td>
<td>• %USERPROFILE%\Application Data\WebClientApps</td>
</tr>
</tbody>
</table>
The lack of Administrator write privileges imposes some limitations on WebClient applications deployed as personal instances. There are common types of files that require Administrator privileges to install, and if you intend to support personal instances of your WebClient application, your application cannot use these types of files. For more information, see the “System Tasks and personal installations” section on page 6–19.

### InstallShield Setup Player add-on and personal installations

The WebClient One-Click Install requires the InstallShield Setup Player to run properly. The InstallShield Setup Player is an ActiveX control provided by Macrovision Corporation that is installed by Internet Explorer as an Add-on. There are applications besides WebClient that might install the InstallShield Setup Player. Once the Add-on is installed, any application can use it for their over-the-Web installs.

If the Add-on is not present, a personal WebClient installation fails. You can check whether or not the Add-on is installed by choosing **Tools → Manage Add-ons → Enable or Disable Add-ons** in Internet Explorer. Look for **InstallShield Setup Player V12** from **Macrovision Corporation** in the dialog that appears. However, this installation requires Administrator write privileges.

If the Add-on is not present, you can correct this deficit before installing WebClient by having an Administrator install the control directly on the end users’ machines. The OpenEdge installation includes the following installer for the Add-on:

```
OpenEdge-install-dir\webinstall\webclient\web_image\SetupPlayer12.msi
```

You can distribute this installer to your end-users. An Administrator can use the following command to silently install and register the Add-on:

```
source-directory\SetUpPlayer12.msi /qn
```

The next time that Internet Explorer runs on the machine, the Add-on is available and a personal WebClient installation can succeed.

If you need to uninstall the Add-on, have an Administrator run this command:

```
MsiExec.exe /X{80482183-337F-46D1-AF8B-F2686C9E7498}
```
Deploying an Application

Choosing an installation method

Before beginning to assemble your application, you must decide whether you want to use the IntelliStream application install feature or an external installer for your WebClient application. You can use IntelliStream to automatically download and update the front-end components of your application. Using IntelliStream, you can have an application installed and updated on an as-needed basis. Alternatively, you can use an external installer to install and update your application. If you want, you can also give your end user the option of updating an application or continuing to use an older but acceptable version of the installed application. For more information about the options available, see Chapter 2, “Designing Your End User’s Experience.”

IntelliStream

When you use IntelliStream, the WebClient Application Assembler generates all the deployment files necessary to install and run your application. In addition, when you make changes to your application, you can use the WebClient Application Assembler to easily generate the files you need to update your application.

You take advantage of the IntelliStream features by defining application components. To define a component, your compiled application code must be available to your development machine. In addition, you must package the compiled application code to reflect how you want to deploy the application. For example, you must add your r-code to any procedure libraries that you include, and all the files must reside in the same directory structure that you intend to create on your end user’s machine. Therefore, you must be able to access these files through the WebClient Application Assembler’s Add Files to Component dialog. For more information about developing your application and compiling your code, see Chapter 5, “Developing the Application.”

Application components

Components are essential to taking advantage of the IntelliStream feature. Components are collections of one or more files grouped together based on related functionality and are put into a compressed .CAB file for deployment. Components are used to deliver the application files and they should meet the following requirements:

- The code organized into a component should be functionally related because the component is automatically installed as a unit to an end user’s machine. Components can be designated as follows:
  - **At Startup** — Installed before the application first runs
  - **As Needed** — Installed when a user runs your application and needs specific functionality
  - **Ask User at Startup** — Installed before the application runs if the user so requests in response to a prompt

- You can have only one **At Startup** component. The **At Startup** component is downloaded before the application runs the first time. This is usually the main code used to start the application and the code required by all users. You identify this unique component by choosing the **At Startup** option when you define the component in the WebClient Application Assembler’s **Component Definition** dialog.
Choosing an installation method

- Components should be self contained so that all procedures that call each other are together. If you have a procedure that is called from many parts of your application, you might want to include this procedure in the **At Startup** component so it is available to all other components.

For more information about components, see, the “Defining applications in WebClient Application Assembler” section on page 6–14, the WebClient Application Assembler help, and Chapter 5, “Developing the Application.”

**System tasks**

System files, such as `.dll` files, or application-specific files that require special processing, such as updates to the registry, must be defined as **System Tasks** on the **Options** tab in the WebClient Application Assembler. Files specified on the **Options** tab are automatically added to the **At Startup** component.

**ABL install procedure**

You can augment the IntelliStream install capabilities with an ABL install procedure. This procedure is specified on the **Options** tab in the WebClient Application Assembler. This procedure, and any code that it requires, must be put in the **At Startup** component. WebClient runs this procedure at the end of its installation tasks. If you use an ABL install procedure, you can also provide an ABL uninstall procedure that runs at the end of the uninstall tasks. For more information about using ABL installers, see the “Options tab” section on page 6–17.

**Uninstall**

IntelliStream includes an installer that removes your application and its registry entries from your end user’s machine.

**External installer**

If you choose to use an external installer that you create yourself, the external installer must complete all the installation tasks required to install your application and must also install all registry keys required by WebClient. You must also provide an uninstaller for your application.

**WebClient registry keys and shortcuts**

When WebClient is installed, it creates registry entries based on the input from the user and the type of installation. Administrator instances use the following registry key:

```
HKEY_LOCAL_MACHINE\Software\PSC\WebClient
```

Personal instances use the following registry key:

```
HKEY_CURRENT_USER\Software\PSC\WebClient
```

These registry key values specify path settings for accessing WebClient functionality, and user-supplied options for running WebClient. You must be aware of these registry entries when you write your application installation. The path settings include two values that you can use to set up application shortcuts and a default setting for your application installation directory. For information on programmatically checking the installation type, see the “Checking application registry access” section on page 5–5.
Table 6–3 lists the registry key value names as they appear in the registry and provides a description of the information that appears under Data in the registry.

**Table 6–3: Registry entries made by WebClient**

<table>
<thead>
<tr>
<th>Registry key value name</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>ProwciniPath</td>
<td>The complete path, including filename, of the WebClient Initializer executable.</td>
</tr>
<tr>
<td>ProwcamPath</td>
<td>The complete path, including filename, of the WebClient Application Manager executable.</td>
</tr>
<tr>
<td>ApplicationsPath</td>
<td>Default path on the end user’s machine that you can use as a root directory to install your WebClient application. The WebClient installation allows the end user to specify this location. Using this registry value in your application installer eliminates the need to prompt the users for information about where they want the application installed. As a result, this creates a more silent application installation. However, you do not want to use this registry value if there might not be enough space on a drive to install the application.</td>
</tr>
</tbody>
</table>

You can provide a shortcut for the user to launch the application directly from Windows after the application is initially installed. This shortcut executes the WebClient Initializer, passing an application configuration file reference to it. You must invoke the application this way to take advantage of the automatic update capabilities of WebClient. When you set up the shortcut in the installer, use the following command syntax to execute WebClient Initializer:

**Syntax**

```
ProwciniPath { "<VendorName>/<ApplicationName>" -s }
```

**ProwciniPath**

The complete path, including filename, to the WebClient Initializer executable. This pathname is stored by the WebClient installer as the ProwciniPath registry key value.

**<VendorName>/<ApplicationName>**

A unique identifier for an application. The values for VendorName and ApplicationName match the values you specify in the General tab of the WebClient Application Assembler when you define your application. WebClient Initializer uses these values to indirectly locate the application configuration file.

**Caution:** When you change the location of your application configuration file, the user’s shortcut no longer works. Tell your users to start the application from the application Web page to automatically update the location of the application configuration file. After updating the location, the shortcut works again.
The WebClient installer also installs a shortcut for running the WebClient Application Manager. You can create a separate shortcut that runs the WebClient Application Manager with your application highlighted. To set up this application-dedicated shortcut, use the following command syntax:

Syntax

\[ \text{ProwcamPath } "<\text{VendorName}>/<\text{ApplicationName}>" \]

**ProwcamPath**

The complete path, including filename, to the WebClient Application Manager executable. This pathname is stored by the WebClient installer as the *ProwcamPath* registry key value.

**<VendorName>/<ApplicationName>**

A unique identifier for an application. The values for *VendorName* and *ApplicationName* match the values you specify in the General tab of the WebClient Application Assembler when you define your application. WebClient Application Manager uses these values to access information about the WebClient Application.

**Application installer registry key entries for vendor and application**

The installer is responsible for setting the Application Directory. The installer either gets the Application Directory information from the *ApplicationsPath* key set by WebClient, or overrides this value by prompting users to indicate where they want the application installed.

If the installer runs successfully, it must store the *ApplicationInstallVersion* so the WebClient initializer knows that it has run and does not need to run again.

To register this information, your installation program must create the following registry key and registry key values for this key:

\[ <\text{Basekey}>\text{\Software}\langle\text{VendorName}\rangle\langle\text{ApplicationName}\rangle\]

Where *<Basekey>* is either the HKEY_LOCAL_MACHINE (HKLM) or the HKEY_CURRENT_USER (HKCU) registry hive depending on the installation type, and *<VendorName>* and *<ApplicationName>* match the values you enter in the General tab of the WebClient Application Assembler when you define your application.
Table 6–4 lists the registry key value names as they appear in the registry and provides a description of the information that appears under Data in the registry.

### Table 6–4: Vendor and application name registry key values

<table>
<thead>
<tr>
<th>Registry key value name</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>ApplicationDirectory</td>
<td>Directory where the user installed the application. This value can be derived from ApplicationsPath.</td>
</tr>
<tr>
<td>ApplicationInstallVersion</td>
<td>Application install version identifier. You must set this value so that WebClient Initializer can determine whether or not your external installer ran successfully. This version must match the version entered in the Options Tab of the WebClient Application Assembler for the WebEnabled Install. For more information, see the “Options tab” section on page 6–17.</td>
</tr>
</tbody>
</table>

Application installer registry key entries for the configuration file

The ProwcappLocator registry keys are required if the installer sets up a shortcut for the application. Your installation program must create the following registry key and registry key values defining the server for the application configuration file under this key:

\[<Basekey>\Software\<VendorName>\<ApplicationName>\ProwcappLocator\]

Where <Basekey> is either the HKLM or the HKCU registry hive depending on the installation type, and <VendorName> and <ApplicationName> match the values you enter in the General tab of the WebClient Application Assembler when you define your application.

Table 6–5 lists the registry key values as they appear in the registry and provides a description of the information that appears under Data in the registry.

### Table 6–5: Configuration file registry key entries

<table>
<thead>
<tr>
<th>Registry key value name</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Indicates the type of server on which the configuration file is located. The type must be InternetServer.</td>
</tr>
<tr>
<td>URL</td>
<td>Well formed URL used to identify where the configuration file is located. The URL must begin with HTTP://, HTTPS://, or File://.</td>
</tr>
<tr>
<td>Filename</td>
<td>Filename of the application CAB file containing the application configuration file (.prowcapc).</td>
</tr>
<tr>
<td>EndUserDescription</td>
<td>Information displayed to your end users when WebClient requests authentication information. For more information see Chapter 4, “Designing Security.”</td>
</tr>
</tbody>
</table>
Choosing an installation method

Application installer registry key entries for WebClient applications

Your installation program must create a registry key value used by the WebClient Application Manager under the following key:

```
<Basekey>\Software\WebClient\Applications\
```

Where `<Basekey>` is either the HKLM or the HKCU registry hive depending on the installation type.

Table 6–6 lists the registry key value name as it appears in the registry and provides a description of the information that appears under **Data** in the registry.

### Table 6–6:  WebClient applications registry key value

<table>
<thead>
<tr>
<th>Registry key value name</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prompt</td>
<td>Indicates whether WebClient should prompt the user for authentication information for the server before downloading the application configuration file. This value can be either <strong>Yes</strong> or <strong>No</strong>. If <strong>Yes</strong>, WebClient prompts the user for authentication information.</td>
</tr>
<tr>
<td>DisablePersistentCache</td>
<td>Indicates whether or not a users can save authentication information in a persistent cache on their machines. This value can be <strong>Yes</strong> or <strong>No</strong>. If <strong>Yes</strong>, the information cannot be saved. If <strong>No</strong>, the information can be saved.</td>
</tr>
</tbody>
</table>

**Application installer registry key entries for WebClient applications**

Your installation program must create a registry key value used by the WebClient Application Manager under the following key:

```
<Basekey>\Software\WebClient\Applications\
```

Where `<Basekey>` is either the HKLM or the HKCU registry hive depending on the installation type.

Table 6–6 lists the registry key value name as it appears in the registry and provides a description of the information that appears under **Data** in the registry.

### Table 6–6:  WebClient applications registry key value

<table>
<thead>
<tr>
<th>Registry key value name</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;VendorName&gt;/&lt;ApplicationName&gt;</code></td>
<td>&quot; &quot;</td>
</tr>
</tbody>
</table>

**Note:** There must be a space between the two quotes.

### Using a combination of IntelliStream and an external installer

You can use a combination of IntelliStream and an external installer for your application. Progress Software Corporation recommends that if you use a combination, you should use an external installer for the initial installation and IntelliStream for updates. If you do use a combination, your installer needs to provide the following registry key entries for the components and the install version of your application.
Application installer registry key entries for components

If you use an external installer for the initial installation of your components, but you want to use IntelliStream to update those components, your installer must store the name of each component it installs in addition to the other tasks required by your external installer. For more information, see the “External installer” section on page 6–7. Therefore, your installation program must create the following registry key and registry key values for each component under this key:

\<Basekey>\Software\<VendorName>\ApplicationName>\Components

Where \<Basekey> is either the HKLM or the HKCU registry hive depending on the installation type, and \<VendorName> and \<ApplicationName> match the values you enter in the General tab of the WebClient Application Assembler when you define your application.

Table 6–7 lists the registry key value name as it appears in the registry and provides a description of the information that appears under Data in the registry.

Table 6–7: Component registry key entry

<table>
<thead>
<tr>
<th>Registry key value name</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;ComponentName&gt;</td>
<td>&quot; &quot;</td>
</tr>
</tbody>
</table>

ComponentName is the identifier you specified for the component in the WebClient Application Assembler Component Definition dialog.

Note: There must be a space between the two quotes.
Providing an uninstall

You must provide an uninstall for your application when you use a combination of IntelliStream and an external installer. Your uninstall must remove all parts of the application installed and all registry entries made by the external installer. However, before performing these tasks, the uninstall must first run WebClient Initializer to remove any updates and registry entries made by IntelliStream. To call WebClient Initializer, use the following command syntax:

**Syntax**

```
ProwciniPath { "<VendorName>/<ApplicationName>"-u } [ -nonadmin ]
```

*ProwciniPath*

The complete path, including filename, to the WebClient Initializer executable. This pathname is stored by the WebClient installer as the *ProwciniPath* registry key value.

*<VendorName>/<ApplicationName>*

A unique identifier for an application. The values for *VendorName* and *ApplicationName* match the values you specify in the **General** tab of the WebClient Application Assembler when you define your application. WebClient Initializer uses these values to locate information about the WebClient application to perform the application uninstall.

*-nonadmin*

This option specifies that you are uninstalling a personal instance. You must include this option to uninstall a personal instance.
Defining applications in WebClient Application Assembler

With the WebClient Application Assembler, you create a WebClient project file (.wcp file), which is a definition of the latest version of an application. After building the project file, you generate a set of deployment files for a version of your application. These generated deployment files are placed in a version-specific directory under an output directory that you specify.

You perform the following tasks with the WebClient Application Assembler:

- Create a new project file
- Modify an existing project file
- Generate all files necessary for installing and updating a WebClient application

Note: You can use the WebClient Deployment Packager to modify details of the deployment configuration, such as the codebase location, rather than regenerating the deployment files. For more information, see the “Customizing deployment configurations” section on page 6–23.

To run the WebClient Application Assembler:


2. Choose WebClient Application Assembler . The WebClient Application Assembler appears:

Note: Alternatively, you can run OpenEdge-Install-Directory\bin\prowcappmgr.exe from a Proenv command prompt.

The remainder of this section describes the WebClient Application Assembler functionality. For detailed information about any options, see the PRO*Tools online help.
General tab

The General tab, shown here, provides basic information about the application, including startup information:

- **Vendor**: Progress
- **Application**: Spotspec
- **Acceptable Version(s)**: 2.0
- **Configuration File URL**: http://spotspec2.91/locators/locators.loc
- **Startup Parameters**: 
  - Insecure: true
  - Insecure: true
  - Insecure: true
- **Application Root Dir**: C:\Program Files\spotspec\samples\Webclient\config

Click **Locator** to open the **Configuration File Locator Definition** dialog. With this dialog, you specify the Internet server for the application configuration file and you define the server’s authentication options:

Click **Versions** to open the **Versions** dialog. With this dialog, you specify which versions of your application are acceptable to run:
Component tab

On the Component tab, you can add, remove, or change components for your application. Use this tab only if you use IntelliStream to install or apply updates for your application:

Click Locator to open the Codebase Locator Definition dialog. Here, you can specify the server of your application components and component updates (.CAB) files:

Click Add or Edit to open the Component Definition dialog. Use this dialog to define your component files and specify the component type:
Options tab

The **Options** tab specifies options for the application installation. On this tab, you can indicate whether you are using an external installer, identify any system tasks you want IntelliStream to perform, and specify an ABL install procedure to handle additional installation steps.

Click **System Tasks Definition** to open the **IntelliStream System Tasks Definition** dialog:

You use the **IntelliStream System Tasks Definition** dialog to define the following system tasks:

- Running ini2reg.exe
- Adding shortcuts on the desktop
- Installing system or common files
- Registering application-specific files, such as the following ActiveX controls: CSCombo, CSSpin, and PSTimer
• Including a .NET Framework installer to support applications using OpenEdge GUI for .NET

• Including run-time versions of the OpenEdge Ultra Controls for .NET

All files specified to perform system tasks are automatically added to your At Startup component.

**ABL install procedure**

You can use an ABL install procedure to perform installation tasks not available with IntelliStream system tasks. The ABL install procedure runs as the last step of an IntelliStream install to complete any application-specific install tasks that IntelliStream did not handle. You must place this procedure, and any procedures that it calls, in the At Startup component.

If you use an ABL install procedure, WebClient passes "<CurrentVersion>, <NewVersion>" as the value of the -param parameter to the procedure. As a result, when the ABL install procedure is run, the -param parameter is added to the startup parameters specified in the Install Startup Params field in the Options tab.

For example, if the last ABL install procedure was Version 4 of the installer, and the next installer is Version 5, the -param value is now "4,5".

When the ABL installer runs for the first time, there is no version for the current installer, as a result, -param has a value of ",<NewVersion>". For example, if the ABL installer is being installed for the first time, and the version for this installer is V1, the -param parameter value is ",V1".

---

**Note:** If you use an ABL install procedure to perform installation tasks, you can also provide an uninstall procedure.

---

**Web-enabled external installer**

Use these options to provide information about your external installer. For more detailed information about this option, see the “External installer” section on page 6–7.

---

**Note:** If you use an external installer, you must provide an uninstall program. You do not need to identify the uninstall program using the WebClient Application Assembler. Your installer needs to make the uninstall program available to your users.
System Tasks and personal installations

Note that certain System Tasks require privileges usually restricted to members of the local or domain Administrators groups. If installing your application requires these tasks, users cannot install personal instances of the application. Such an application must use an Administrator install.

For example, certain files get installed to directories to which only an Administrator can write. The **IntelliStream File Definition** dialog flags these file types, as shown:

![IntelliStream File Definition](image)

If your application uses the OpenEdge GUI for .NET, the user’s machine requires an appropriate version of the .NET Framework. Only an Administrator can install the framework, so the **.NET** tab flags the options for including a .NET Framework installer, as shown:

![IntelliStream System Tasks Definition](image)

For more information on personal installations and .NET, see the “WebClient and OpenEdge GUI for .NET” section on page 6–44.
WebClient tab

The **WebClient** tab provides information about installing WebClient for your application:

<table>
<thead>
<tr>
<th>Component</th>
<th>Options</th>
<th>WebClient</th>
</tr>
</thead>
<tbody>
<tr>
<td>WebClient Support for Your Application</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WebClient Install URL</td>
<td><a href="http://www.progress.com">http://www.progress.com</a></td>
<td></td>
</tr>
<tr>
<td>WebClient Version(s)</td>
<td>R10.2A</td>
<td></td>
</tr>
</tbody>
</table>

**Note:** If your application uses the OpenEdge GUI for .NET, the **WebClient Version(s)** field must include only releases that support the OpenEdge GUI: OpenEdge R10.2A and later.
Files generated by the WebClient Application Assembler

After you define the installation requirements for your application and you are ready to deploy, use the WebClient Application Assembler Generate dialog to create the deployment files for a version of your application. To access the Generate dialog, choose Deployment → Generate... or click Generate on the toolbar. The Generate dialog appears:

Whenever you use the Generate dialog to create a version of your application, the WebClient Application Assembler creates a directory to store the deployment files for the new version of the application. The directory name matches the value of the Version Name and is a subdirectory of the Output Directory. The name for this version directory appears in the Version Directory.

When you regenerate a version of the application, the WebClient Application Assembler first deletes all the files in the version directory and then regenerates the deployment files.

Caution: You must not regenerate a version of an application that you have already deployed.

When you generate a version of your application, the WebClient Application Assembler validates and updates the information about your application and creates the following files:

- **Application configuration file** — Contains deployment information for your WebClient application. The configuration filename is derived from the name of the project file and has an extension of .prowcapp. The WebClient Application Assembler places this file in a .CAB file that has an extension of .prowcapc. For example, if your .wcp filename is AutoInsurance.wcp, the WebClient Application Assembler places a file named AutoInsurance.prowcapp into a .CAB file named AutoInsurance.prowcapc.

  If you click Security and select a digital signature option in the Security window, the .prowcapc file and all the component files are signed with a digital certificate. (See the “Defining an application as signed” section on page 4–3 for more information.)
Deploying an Application

- **WebClient version information file (.wcv)** — Contains version information for all the files within the application, including .r files within procedure libraries. A .wcv file is generated for each version and is saved in the version directory.

  **Note:** The .wcv file is not deployed.

- **Application component files (.CAB)** — Contain the files in an application component. The WebClient Application Assembler generates one .CAB file for each component and saves it in the version directory. The root of the .CAB filename is the component name you specify in the Component dialog.

- **Application components update files (.CAB)** — Contain the files necessary to update the application components. The WebClient Application Assembler generates one .CAB file for each component that needs updating, and saves it in the version directory. Each .CAB file contains the changed files and a .wcm file that provides instructions for applying the changes to update the application. Each .CAB file has a unique name based on the component name, the previous version, and the new version.

  **Note:** If you use an external installer, you must prepare this install image yourself.
Customizing deployment configurations

Users of your application might have specific requirements for how it is deployed. Often you cannot be aware of these requirements at the time that you generate the .prowcapc file. Even when custom requirements are known, maintaining separate development streams to support deployment to multiple users can be burdensome and error-prone. The WebClient Deployment Packager enables you to address this problem without changing the .wcp project file and regenerating the entire application. With the Deployment Packager, you can:

- Extract deployment information from an existing .prowcapc file.
- Modify the deployment information and save the changes to a file referred to as a custom deployment configuration. This file has a .wcd extension and the same name as the .prowcapc file (for example, a custom deployment configuration extracted from sportspro.prowcapc is stored as sportspro.wcd).
- Generate a new .prowcapc file incorporating the changes made to the .wcd file.

You might use this capability in the following cases:

- Supporting customers with different code pages or languages
- Providing access to a test environment in addition to your production environment
- Customizing deployments that each customer hosts internally

After the initial generation of the .prowcapc file with the WebClient Application Assembler, the developer or the customer can use the Deployment Packager to modify the configuration details as required. OpenEdge provides a standalone Deployment Packager executable that you can make available to customers, enabling them to customize their own configurations.

Distribution of the Deployment Packager

The Deployment Packager is an integral part of the PRO*Tools suite and is installed with OpenEdge Architect. End users can also create their own custom deployment configurations with the Deployment Packager by working with the .prowcapc files that they receive from application developers. Since end users typically do not have access to OpenEdge Architect, a standalone version of the tool and the required libraries is provided in a file named prowcdp.zip. You can distribute this version to your customers as appropriate.

Distributing and installing the standalone version

The prowcdp.zip file is installed in the OpenEdge bin directory. You can copy and distribute this single file to customers and other users. On the target system, extract the contents of the archive into a single directory. Then, in that directory, execute the following commands to register components:

```
regsvr32 psccab.dll
regsvr32 psccert.dll
regsvr32 pscsecreg.dll
```

You should see a success message after each command. The Deployment Packager is now ready for use.
Changes allowed through the Deployment Packager

The Deployment Packager can change the following elements of the application definition:

- Application name
- WebClient install URL
- Configuration file URL and associated security information
- Codebase URL and associated security information
- Digital signature information for the .prowcapc file
- URLs for Web-enabled external install and upgrade programs

You cannot use the Deployment Packager to modify the content of the application itself.

Defining a custom deployment configuration

Before running the WebClient Deployment Packager, determine the correct values for your custom deployment configuration, and make sure that the source .prowcapc file is accessible.

To run the WebClient Deployment Packager:

1. Choose **WebClient Deployment Packager** from the PRO*Tools palette; or execute the following command from either `<OpenEdge-install-dir>\bin`, or the directory where you unzipped prowcdp.zip:

   ```
   prowappmgr.exe -deploy
   ```

   The **WebClient Deployment Packager** window appears:
2. Create a new custom deployment configuration, or open an existing one.

3. You can save the deployment configuration to a .wcd file at any time. If the configuration has not been saved previously, you see a standard **Save As** dialog that enables you to name the file with a .wcd extension and to specify its location.

4. Most of the Deployment Packager fields get their initial values from the source .prowcapc file. Enter new values as appropriate for the custom deployment configuration. See the “Defining applications in WebClient Application Assembler” section on page 6–14 and the online help for more information.

---

**Caution:** Your .prowcapc file and your application .CAB files must be either both **signed** or both **unsigned**. The application will not run if there is a mismatch. Therefore, you must be sure that the security setting for your new .prowcapc file matches that of the .CAB files. The value that you specify in the **Current** security field applies only to the new .prowcapc file; it does not affect the .CAB files.

The value (**Signed** or **Unsigned**) that the Deployment Packager shows for the **Original** .prowcapc file typically reflects the status of the .CAB files, as well as that of the source .prowcapc file, so you should usually match that setting. If the security settings do not match when you validate or generate the new configuration, a warning is displayed.

5. After defining the custom deployment configuration, save your changes and generate the new .prowcapc file, as explained in the “Generating the .prowcapc file” section on page 6–26. Optionally, you can validate the configuration in a separate operation before generating the new file. (Before executing a Generate operation, the Deployment Packager automatically performs the same validation.)

---

**Security considerations**

Customers who host WebClient applications developed by an outside application provider must coordinate with the provider to make sure security is properly configured. These customers also must instruct their end users to recognize and trust their digital signatures.

If the application provider makes the prowcdp.zip file available, the customer can run the Deployment Packager and manage the security settings and other settings independently. Alternatively, the customer may specify the requirements and ask the application provider to define the deployment configuration accordingly.

Working as necessary with the application provider, the customer must make sure that:

- Both the .prowcapc file and the .CAB files are signed if digital signatures are used. (You do not need to use the same certificate for both; either the customer or the provider can sign the files.) If one is signed and the other is unsigned, the application will not run. You can only specify the digital signature (or the absence of it) for the .prowcapc file with Deployment Packager. Typically, the application provider must make any required changes to the .CAB files.

- The authentication settings, as well as the URLs, are correctly specified for the Configuration File Locator and the Codebase Locator. Either the customer or the application provider can adjust these settings through the Deployment Packager.
Deploying an Application

- End users are properly informed how to respond to security prompts. For digitally signed files, you should instruct end users to trust content signed by either the customer or the application provider, as applicable.

Validating the custom deployment configuration

To check for problems before generating a new .prowcapc file, open the .wcd file and choose Deploy→Validate. The Deployment Packager analyzes the settings you specified and reports any of the following that it detects:

- **Error conditions** — Problems that you must correct before generating the new .prowcapc file, such as invalid or missing values in the Prowcapc File, Application Name, and Output Directory fields.

- **Warning conditions** — Settings that might cause problems when the application runs, but that do not prevent generation of the new .prowcapc file. For example, it displays a warning if the new application name differs from the name of the source .prowcapc file.

If there are no errors, the Deployment Packager displays the “Validation succeeded” prompt that lists the changes you made—that is, differences between the original configuration and the new one—as well as any warnings. Review the information before dismissing this prompt.

If errors are found, the Deployment Packager lists them in the “Validation failed” prompt. You must correct the errors before the Deployment Packager can generate the new file.

Generating the .prowcapc file

When you are ready to store your custom deployment configuration in a new .prowcapc file, you can do so in the following modes:

- **Interactive mode** — Starts the generation process from the Deployment Packager main window.

- **Batch mode** — If OpenEdge Architect is installed, you can execute a command-line script that performs the operation without user interaction.

In both cases, the Deployment Packager writes a log of the operation in the output directory where it creates the .prowcapc file. The log file has the same name as the .prowcapc file, but with a .log extension.

**Interactive mode**

Open the .wcd file and choose Deploy→Generate, or click on the toolbar. The Deployment Packager validates the configuration, as described in the preceding section.

If there are no errors, the Deployment Packager displays a prompt that lists the changes you have made—that is, differences between the original configuration and the new one—as well as any warnings. At this point, you have the option of continuing or canceling the operation. After reviewing the information, click Yes to generate the new .prowcapc file, or click No to abort the operation.
When you click Yes, the Deployment Packager generates the new .prowcapc file in the specified output directory. The filename matches the value of the Application Name field.

Upon completion, a message identifies the new .prowcapc file (or indicates the errors that prevented successful generation of the file). It also identifies the log file for the operation.

**Batch mode**

If OpenEdge Architect is installed, you can use the bprowcappdep.bat script in `<OpenEdge-install-dir>\bin` to generate the .prowcapc file.

**Note:** OpenEdge Architect is not available to users of the standalone Deployment Packager tool extracted from the prowcdp.zip file.

You execute the script passing it the path to the deployment configuration (.wcd) file, for example:

```
bprowcappdep c:\WebClient\deployment\MyApp.wcd
```

Unless the validation process encounters errors (in which case the operation terminates), generation of the .prowcapc file proceeds with no option to cancel. Messages that precede the actual start of the generation process, including error notifications and warnings, are written to the console window; they do not appear in the log file. Therefore, when using the bprowcappdep.bat script, you should redirect the screen output to a file or take other steps to make sure you do not overlook important messages.
Hosting the application on a server

After you use the WebClient Application Assembler to generate your deployment files, you must place these files on your deployment server so that WebClient can access the files.

You have some options when deciding on the type of server you want to use to host your application. You can use an Internet-based Web server or file server, or you can use an AppServer. For more information about design considerations to help you make these decisions, see Chapter 3, “Hosting the WebClient Deployment.”

This section also provides information about:

- Hosting the configuration file
- Hosting your application component files
- Hosting your external installer
- Hosting on a Web server

Caution: For all server types, when copying files from your development machine to another machine, you must use a binary copy for all files.

Hosting the configuration file

Place the application configuration file in the directory specified by the URL in the WebClient Application Assembler’s Configuration File Locator Definition dialog.

Note: The application configuration file cannot be hosted on an AppServer.

The first time the end user downloads the configuration file, the bootstrap.htm file specifies its location. For more information about this bootstrap.htm file, see the “Bootstrapping a WebClient application” section on page 6–36. When downloaded, the configuration file is cached on the user’s machine.

Although the application configuration file is generated in the version directory, you should place the file in a version-independent directory for deployment. You must always deploy the application configuration file to this same directory, regardless of the version number.

Caution: If you move the configuration file, you must change all references to the location of this file. This includes the references in the bootstrap.htm and webclient.htm files, and the URL specified in the Configuration File Locator Definition dialog. You must then use the WebClient Application Assembler to generate a new version of the application configuration file. Also, if you provided a shortcut for running the application, you must tell your users to rerun the bootstrap process from the initial installation Web page to automatically update the shortcut.
Hosting your application component files

Place the component (.CAB) and component update files for your application in the directory that you specified in the WebClient Application Assembler’s Configuration File Locator Definition dialog. For more information about specifying a codebase locator, see Chapter 3, “Hosting the WebClient Deployment.”

You should copy the version directory and its contents, including all its subdirectories, as specified in the dialog. The application version directory on the server machine must match the the version directory used in the WebClient Application Assembler Generate dialog.

Where you can place your application files depends on which type of server you use:

- **AppServer** — The application version directory must be relative to the AppServer’s working directory.
- **Web server** — The application version directory must be relative to the root directory of your Web server.
- **File server** — The application version directory should be a subdirectory of the directory specified in the file URL.

Hosting your external installer

If you use an external installer for installing or updating your application, you must do the following to deploy the external installer:

- If you use an external installer to install your application, you must copy the installer to the location you specified in the Install URL field of the WebClient Application Assembler’s Options tab.
- If you use an external installer to update your application, you must copy the installer to the location you specified in the Update URL field of the WebClient Application Assembler’s Options tab.

For more information on specifying these values, see the “Options tab” section on page 6–17.

Hosting on a Web server

Deploying your application component files or external installer on a Web server presents extra considerations. The following sections provide information about:

- Case sensitivity on UNIX Web servers
- Executable directories
- Authentication options on Microsoft Internet Information Server (IIS)
- Configuring MIME types for your Web server

**Case sensitivity on UNIX Web servers**

Web servers that use the UNIX operating system, such as Apache, are case-sensitive. Mismatched cases between the installation filenames on the Web server and those in your application configuration file will cause a run-time error.
**Executable directories**

Do not place a Web installer in an executable directory on your Web server. Otherwise, when a user’s browser accesses the installer executable, the installer attempts to execute on the Web server instead of downloading and executing on the user’s system. This results in a server run-time error. See your Web server documentation for information on how to plan the content of your directories (or to create virtual directories) so they do not allow local execution of files referenced by the user’s browser.

**Authentication options on Microsoft Internet Information Server (IIS)**

If you are configuring Microsoft IIS, select only the Basic (Clear Text) authentication option in the WWW Services Properties for the Web server. If you select the Windows Challenge/Response authentication option, end users who are not running Internet Explorer receive an authentication dialog that they cannot satisfy, preventing them from starting an installation. This prevents end users from using any other Web browser with your application.

**Note:** Basic authentication uses a well-known encryption algorithm for user names and passwords that provides minimal security.

**Configuring MIME types for your Web server**

When the user attempts to access the application, the Web page that you provide references the configuration file for the application. Before your user’s browser can open the configuration file using WebClient Initializer, it must recognize a MIME type for the configuration file. To accomplish this, you must configure your Web server to associate the configuration file extension with its MIME type.

The method used for configuring your Web server depends on the type of Web server you use. For each Web server, however, you must specify the extension and MIME type for the configuration file types, as follows:

- **MIME types** — application/progress-wcappcab
- **File type extension** — prowcapc

**Note:** If an Internet Service Provider (ISP) hosts your Web installation, you must give them the information to configure the MIME type properly on their Web server.
The following procedures cover configuring the MIME type on common Web servers.

To register the MIME types for all directories on an Apache Web server, do one of the following:

- Add the following line to the `srm.conf` file:

  ```
  addtype application/progress-wcappcab prowcapc
  ```

- Add the following line to the `mime.types` file:

  ```
  application/progress-wcappcab prowcapc
  ```

If you are using a dedicated Apache Web server, rehash/recycle to register this information.

For a Sun Java System Web Server, you must change the association of `.exe` files in addition to adding the definition for `application/progress-wcappcab`.

To configure MIME types on Sun Java System Web Server:

1. Open the Administration Server.
2. Select a server and choose the Manage button.
3. Choose Preferences → Mime Types.
4. Add the new content-type definition `application/progress-wcappcab`, and associate it with the file suffix `prowcapc`.
5. Edit the `magnus-internal/cgi` MIME type definition, and remove `exe` from the list of associated extensions.
6. Edit the `application/octet-stream` MIME type definition, and add `exe` to the list of associated extensions.
7. Save and apply your changes so the Web server recognizes the changes to the MIME type definitions.
The Microsoft IIS server stores its MIME types in the Windows registry. You register the MIME type for the application configuration file using the \texttt{iis\_wc.reg} registry file found in \texttt{<OpenEdge-install-dir>\webinstall\webclient\web\image}.

To merge the MIME type entries from \texttt{iis\_wc.reg} into the registry and configure the Web server:

1. Copy the \texttt{iis\_wc.reg} file to your Web server.
2. Register the MIME type by doing one of the following on the Web server:
   - Select \texttt{iis\_wc.reg} in the Windows Explorer, right-click, and select \texttt{Merge}. This imports the contents of the file into the registry.
   - Double-click the file in the Windows Explorer. This imports the contents of the file into the registry.
3. Restart your Web server to register these changes.
Installing WebClient

You must make an installer available to install WebClient on the user’s machine. OpenEdge provides a default InstallShield image that you can customize to install WebClient. You can also point the user to a page on the Progress Software Corporation Web site where the user can run a WebClient installation.

Your OpenEdge installation includes the following WebClient install images:

- A stand-alone InstallShield image that you can use to create an installation CD-ROM.
- A One-Click Install image that you can use to host a Web-enabled installation.

For more information about copying theses install images to your system, see the “Accessing the WebClient installation files” section on page 6–33.

For a Web-enabled install, you can set up a WebClient installation Web page to be one of the following:

- An OpenEdge-supplied Web page (.webclient.htm file). You can customize this file to start your application installation after WebClient is installed.
- An application-specific Web page of your own design. The logic for this page should run the WebClient One-Click Install directly, rather than bringing up the default WebClient One-Click Install Web page (which requires the user to click another button). To do this, you can copy the logic from the .webclient.htm file to your own Web page.

Though you cannot customize the CD-ROM image, Progress Software Corporation strongly recommends you customize the One-Click Install. For more information, about customizing the WebClient installation, see the “Customizing the WebClient installation” section on page 6–34.

Note: Whatever Web page you provide for accessing the WebClient installer, you must specify the URL for this page in both the bootstrap.htm file and in the WebClient Install URL field of the WebClient Application Assembler’s WebClient tab.

Accessing the WebClient installation files

The prebuilt WebClient InstallShield image is a ready-to-deploy WebClient installer that you can find in <OpenEdge-install-dir>\webinstall\webclient\web_image.

To use this installer, do either of the following:

- To create a CD-ROM for installing WebClient, copy the web_image directory contents to the CD that you want to distribute.
- To provide a Web installer for installing WebClient over the Web, copy the web_image directory contents to an appropriate location on your Web site.

Although you can deploy this installer without customization, Progress Software Corporation recommends you customize it to install your WebClient application along with WebClient. If you provide a prebuilt WebClient installer without customization, you must instruct your end users to install your WebClient application as a separate installation step. For more information, see the “Customizing the WebClient installation” section on page 6–34.
Customizing the WebClient installation

To customize a Web-enabled WebClient installer, you modify certain lines in the default One-Click Install Web page to run WebClient Initializer or to invoke your application installer directly if you are using an external installer. Running WebClient Initializer leads your user through the next installation step to automatically install your WebClient application.

**Caution:** Any modifications that you make to the WebClient One-Click Install must not interfere with the installer’s original operation. You can only add functionality to the operations that it performs by default, and change the appearance of the page. Any other changes might produce unpredictable results in the WebClient installation.

You can find a copy of the default WebClient Web page (webclient.htm) in the following OpenEdge installation location:

```
OpenEdge_install_dir\webinstall\webclient\web_image\webclient.htm
```

to implement one of the supported customizations, you must modify the **bolded** portions of the following webclient.htm fragment:

```javascript
function startInstall(){
    // TODO: To run the WebClient Initializer after the WebClient install completes, you
    // must provide the location (including the file name) of your prowcapc file. To do this,
    // remove the "//" at the beginning of the last line of this comment (to uncomment it).
    // Then customize that line by replacing <URL...> with the location and name of your
    // prowcapc file as in this example:
    //
    // ether.SetProperty("ProwcappURL",
    // "http://server1/webclient/app_installer/yourapp.prowcapc");
    //
    // ether.SetProperty("ProwcappURL", "<URL to your .prowcapc file>");
    // OR: If you are using a One-Click application install (implemented using InstallShield
    // V6 or later) you can run it directly after the WebClient install completes by
    // providing the location (directory only) of your installer image plus the location
    // (including the file name) of your prowcapc file. To do this, remove the "//"
    // at the beginning of the last 2 lines of this comment (to uncomment them).
    // Then customize those lines by replacing <URL...> with the directory
    // containing your application installer image and the location and name of your
    // prowcapc file as in this example:
    //
    // ether.SetProperty("ApplicationURL", "http://server1/webclient/app_installer");
    // ether.SetProperty("ProwcappURL",
    // "http://server1/webclient/app_installer/yourapp.prowcapc");
    //
    // ether.SetProperty("ApplicationURL", "<URL to your One-Click app installer directory>");
    // ether.SetProperty("ProwcappURL", "<URL to your .prowcapc file>");

    ether.LegacyMode = true;
    ether.Play();
}
```
To run WebClient Initializer after the WebClient install, you might customize the `startInstall()` function to look like this:

```javascript
function startInstall(){
    ether.SetProperty("ProwcappURL", "http://NSSWeb:81/webclient/BakeWare_installer/BakeWareV10.prowcapc");
    ether.LegacyMode = true;
    ether.Play();
}
```

This customization invokes the Initializer and passes it the specified `BakeWareV10.prowcapc` file, which will perform the task necessary to install the application.

To directly run your application One-Click Install, if you have one, from the WebClient install, you might customize the `startInstall()` function to look like this:

```javascript
function startInstall(){
    ether.SetProperty("ApplicationURL", "http://NSSWeb:81/webclient/BakeWare_installer");
    ether.SetProperty("ProwcappURL", "http://NSSWeb:81/webclient/BakeWare_installer/BakeWareV10.prowcapc");
    ether.LegacyMode = true;
    ether.Play();
}
```

This customization first executes your application’s One-Click Install from the specified `BakeWare_installer` directory on your Web server. It also passes the location of the application configuration file (the specified `BakeWareV10.prowcapc` file) to the application installer. The application installer can then invoke WebClient Initializer to launch the installed application after completing the application installation.
Bootstrapping a WebClient application

Whether you use IntelliStream or an external installer to install your application, the WebClient
Initializer always starts your WebClient application. For information about installation options,
see Chapter 2, “Designing Your End User’s Experience.”

The WebClient Initializer controls the installation, the update, and the execution of a WebClient
application. It can also include installing a different version of WebClient itself.

WebClient Initializer uses information in an application configuration file to control how it
starts a WebClient application. The information that you provide in the application
configuration file allows WebClient Initializer to identify and verify the status of both
WebClient and the application on the user’s system, and to start the application.

Starting the application

You can provide two basic options for the user to start a WebClient application:

- **URL** — This option allows the user to start the application from a Web browser. The
  specified URL identifies a **bootstrap.htm** file provided by OpenEdge that you must
  customize to access both the WebClient One-Click install image and your application
  configuration file. This **bootstrap.htm** file starts the process of installing and executing
  both WebClient and your WebClient application. You can make this URL available to the
  user in any way appropriate for your Web site. For example, you can have a user enter the
  URL directly in the locator of their browser, or you can provide a link to this URL on your
  application Web page.

  **Note:** If WebClient is installed from a CD, you should still use the **bootstrap.htm** file to
  install the application.

- **Shortcut** — This option allows the user to start the application directly from the Windows
desktop. If you use IntelliStream, this shortcut can be created automatically using the
System Tasks option in the WebClient Application Assembler. If you use an external
installer, your installer must create this shortcut.

Bootstrapping and personal instances

Personal instances require OpenEdge 10.2B or later. To prevent the WebClient installer from
starting a personal install without an appropriate version of the WebClient, the **bootstrap.htm**
file contains a new variable, `supportPersonalInstall`. This variable defaults to `FALSE`. If your
WebClient application supports personal installs, you must set this variable to `TRUE`. This forces
an early check that there is an appropriate version of WebClient.
How the bootstrapping process works

The prescribed way to bootstrap your WebClient application is to use a customized version of the bootstrap.htm file that OpenEdge provides. The bootstrap.htm file initiates the installation process. For information about customizing this file, see the “Customizing the bootstrap.htm file” section on page 6–39.

The bootstrap.htm file has the following logical paths:

- If WebClient is not installed, your end user is taken to a Web page to install WebClient.
- If WebClient is installed, WebClient Initializer runs and performs the following tasks:
  - Checks that an acceptable version of WebClient is installed on the user’s system and gives the user an opportunity to update the version of the WebClient installation, if necessary.
  - Checks whether the application is installed on the user’s system and gives the user an opportunity to install it, if necessary.
  - Checks whether any application updates are required and gives the user an opportunity to update the application, if necessary.
  - Starts the application.

**Note:** During this bootstrapping process, if any URL requires authentication on your Web server, the end user receives a prompt for the appropriate user ID and password.

Whenever WebClient Initializer runs, it takes your users to the next step of the bootstrapping process. Therefore, any process which the Initializer launches to perform an installation step (WebClient install and external application install) must run the Initializer again for the bootstrapping process to continue, and to ultimately run the application.
Figure 6–1 illustrates the bootstrapping process and shows the bootstrap process flow depending on Initializer’s verification at the different points in the process. This process is transparent to the end user, varying slightly depending on the installation method and any customizations that you made to install and start your application.

![Diagram of the bootstrapping process](image)

**Figure 6–1: Bootstrapping process**

**Note:** To concentrate on other details, Figure 6–1 does not show the processing associated with installation types. For more information on how installation types affect the startup process, see the “Installation precedence” section on page 1–7.
Bootstrapping a WebClient application

Bootstrapping with no WebClient installed

If WebClient is not found on the user’s machine, the bootstrap.htm file opens a WebClient installation page based on a URL that you specify. You must specify the WebClient installation URL and customize the bootstrap HTML file accordingly. For more information on the WebClient installation page, see the “Installing WebClient” section on page 6–33.

Bootstrapping with some version of the WebClient installed

If any version of WebClient is found on the user’s machine, the bootstrap.htm file references a URL to the application’s configuration file, which runs WebClient Initializer. You must specify the configuration file URL and customize the bootstrap.htm file accordingly.

Note: If you set supportPersonalInstall to TRUE and the user’s machine does not have an appropriate version of WebClient installed, the bootstrap process forces an install before running WebClient Initializer.

Customizing the bootstrap.htm file

If you use the bootstrap.htm file, you must customize it. To customize the bootstrap.htm file, you substitute your own URLs for the two default URL settings:

- **document.locations.href** — This setting specifies the URL of the .CAB file containing the application configuration file for the application.

- **window.location** — This setting specifies the URL of the Web page used to access the WebClient installer over the Web. You can use your own customized version of WebClient or the default WebClient installer provided on the Progress Software Corporation Web site.

The option for supporting personal instances and the default URL settings are **bolded** in this sample bootstrap.htm file.

Sample bootstrap.htm file

```html
<html>
<script language="JavaScript">
    installed = false;
    // NOTE: Set supportPersonalInstall to true to require a version of WebClient that supports personal instances,
    // even if there is already a version of WebClient installed
    supportPersonalInstall=false;
    netscape = (navigator.appName.indexOf("Netscape") != -1);

    // See if WebClient is installed
    if (netscape)
    {
        var mimeTypes;
        if (supportPersonalInstall)
            mimeTypes = navigator.mimeTypes["application/progress-wcappna"]; //application/progress-wcappcab"
        else
            mimeTypes = navigator.mimeTypes["application/progress-wcappcab"]; //application/progress-wcappcab"
        if (mimeTypes)
            installed = true;
    }
</script>
```

(1 of 3)
<script language="VBScript">
' This is ignored by Netscape

On Error Resume Next

set ctrl = CreateObject("ProWCVer.ComObj.1")
if isObject(ctrl) then
  if (supportPersonalInstall) then
    installed = supportsNonAdmin(ctrl.Version)
  else
    installed = true
  end if
end if

' Check if the version of WebClient supports personal installs
Function supportsNonAdmin(verstring)
  tmp = false
  cdot1 = InStr(verstring,"."
  if cdot1 > 0 then
    cmajor = Cint(Mid(verstring,1,cdot1 - 1))
    cdot2 = InStr(cdot1 + 1,verstring,"."
    if cdot2 > 0 and cmajor >= 10 then
      tmp = true
    else
      cminor = Cint(Mid(verstring,cdot1 + 1,1))
      cmaint = Mid(verstring,cdot1 + 2,1)
      if cmajor > 10 then
        tmp = true
      elseif cmajor = 10 then
        if cminor < 2 then
          tmp = false
        elseif StrComp(cmaint,"B") < 0 then
          tmp = false
        else
          tmp = true
        end if
      end if
    end if
  else
    supportsNonAdmin = tmp
  end if
End Function
</script>

<script language="JavaScript">
if (installed)
{
  // Reference a prowcapp file to invoke prowcini.exe
  // NOTE: MODIFY THIS URL TO POINT TO YOUR .prowcapc FILE.
  document.location.href = "http://MyWebServer/webclient/myapplication.prowcapc";
}
else
{
  // Start install process for Web Client
  // NOTE: MODIFY THIS URL TO POINT TO THE WebClient Install page
  window.location = "http://MyWebServer/webclient/webclient.htm";
}
</script>
Checking for an acceptable version of WebClient

The WebClient Initializer first verifies that an acceptable version of WebClient is installed on the user’s system. The Initializer compares the installed versions of WebClient with the acceptable versions listed in the application configuration file. If there is an acceptable but older version, the Initializer gives the user an opportunity to install the preferred version or to run with an existing version. If there is no older but acceptable version, the Initializer gives the user the opportunity to install the preferred version.

For any case where the user chooses to install the preferred version of WebClient, the Initializer opens a Web browser on the user’s system to the URL of the WebClient installation specified in the application configuration file and exits, allowing the user to continue with the installation over the Web.

If the user chooses to use an installed acceptable version, the Initializer proceeds with the next validation. Otherwise, the Initializer exits without proceeding.

Whatever WebClient version the user chooses to run, the user can activate a Don’t ask me this again toggle box to prevent notification of another update until you specify a different preferred version in the configuration file.

If WebClient Initializer finds that the application specified in the configuration file is not installed, one of the following happens:

- If you use IntelliStream, the Initializer starts WebClient to install and start the application.
- If you use an external installer, the Initializer opens the user’s Web browser to the URL of the application install page and exits.

Checking for required application updates

If WebClient Initializer finds that the installed version of the application is out of date, it gives the user an opportunity to update it. For more information, see the “WebClient application updates” section on page 6–42.
WebClient application updates

When you make changes to your application, you simply generate a new version of the application with the WebClient Application Assembler. This creates a new configuration file and any necessary component files and component update files, placing them in a new version subdirectory.

On the end user’s machine, WebClient Initializer compares the installed version of the application with the acceptable version in the application configuration file and then prompts the end user as follows:

- If the installed version is acceptable but older than the current version, the Initializer gives the user an opportunity to update to the current (preferred) version or run with the existing version.

- If the installed version is not acceptable, the Initializer gives the user an opportunity to update to the preferred version.

If you use:

- IntelliStream, and the end user chooses to update, WebClient Initializer starts WebClient to perform the IntelliStream update and run the application.

  Depending on how you implemented your security, the end user might be prompted for authentication information and verification to download the modified files. For more information, see Chapter 4, “Designing Security,” and the “Application components” section on page 6–6.

  **Note:** A major benefit of IntelliStream is that only the individual modified files are downloaded during an update. The WebClient Application Assembler determines what files are modified according to an algorithm. To optimize this process, Progress Software Corporation strongly recommends you use the Generate MD-5 option when you compile your r-code. For more information, see the “Determining if a resource was modified” section on page 1–13.

- An external installer to update the application, you can re-use the Web page and logic for initially installing your application, or you can provide a separate application-specific Web page and logic.

For any case where the user chooses to update the application to the preferred version, the Initializer opens a Web browser on the user’s system using the application update URL specified in the application configuration file, and then exits. At this point, the user can continue with the update over the Web.
### Updating procedure libraries

If your WebClient application uses procedure libraries, they might require special handling to update. While IntelliStream can perform incremental updates to standard procedure libraries, the technology does not support incremental updates to memory-mapped procedure libraries.

Progress Software Corporation recommends that you do not use memory-mapped procedure libraries in WebClient applications. If your WebClient application must use memory-mapped procedure libraries, you can consider the following alternatives:

- Do not attempt incremental updates of the memory-mapped procedure library. Instead, whenever the contents of the procedure library change, create a new version of your WebClient application and do not support previous versions with the older procedure library.

- Do not use IntelliStream to update your WebClient application. Use another solution for upgrades.

- Include the standard procedure library from which you create the memory-mapped procedure library in your deployment. Update the contents of the standard procedure library with IntelliStream. Then, use a procedure that creates a new memory-mapped procedure library on the client with PROLIB.
When deploying applications that use the OpenEdge GUI for .NET with WebClient, there are additional issues to consider. These include handling the following:

- .NET Framework
- .NET assemblies
- Other files

**.NET Framework**

Applications using the OpenEdge GUI for .NET require an appropriate version of the .NET Framework to run. An end user cannot start an OpenEdge GUI application without having an appropriate version of .NET installed. When you use WebClient to deploy one of these applications, your deployment must accommodate this requirement.

**Note:** Because users without Administrator privileges cannot install the .NET framework, you should consider this requirement carefully if your WebClient application supports personal installations.

You can address this requirement in the following ways:

- Include the OpenEdge-certified version of the .NET Framework in your WebClient deployment through the **IntelliStream System Tasks Definition** dialog:

  ![IntelliStream System Tasks Definition dialog](image)

  This option packages a .NET Framework installer in the initial WebClient install. The WebClient Initializer checks the end-user’s machine for either that version of the .NET Framework or a later version. If one is found, it aborts the install. This result enables both Administrator and personal installations to continue.
Otherwise, it launches the .NET Framework installer. An end-user with Administrator privileges must accept the Microsoft EULA and complete the .NET Framework installation to continue installing your application. However, the framework installer fails for a non-Administrator, and the WebClientInitializer displays an error and aborts the personal installation.

- Include a user-defined version of the .NET Framework in your WebClient deployment through the IntelliStream System Tasks Definition dialog. This option packages a .NET Framework installer that you supply in the initial WebClient install. Unlike with the OpenEdge-certified version, the WebClientInitializer does not check the end-user’s machine for an existing .NET Framework. The WebClientInitializer always launches the user-defined .NET Framework installer. An end-user with Administrator privileges must accept the Microsoft EULA and complete the .NET Framework installation. However, the framework installer fails for a non-Administrator, and the WebClientInitializer displays an error and aborts the personal installation.

For more information on appropriate versions of the .NET Framework, see the chapter on maintaining user environments in *OpenEdge Deployment: Managing ABL Applications*.

**Caution:** Some .NET Framework installers might uninstall existing versions of the framework. You should verify what a particular installer does before including it in your WebClient deployment.

- Do not include a .NET Framework installer in your WebClient deployment. This option leaves it to the end users to ensure that an appropriate version of the .NET Framework is available on their machines before they install your application.

Because WebClient cannot determine what other applications might require the framework, uninstalling WebClient or your WebClient application does not uninstall the .NET Framework.

**.NET assemblies**

An *assembly* is a .NET library or collection of classes. The WebClient Application Assembler can handle assemblies like other application files, enabling you to package them into components. Your OpenEdge GUI application might use the following types of assemblies that you need to include in your deployment:

- Optional OpenEdge Ultra Controls for .NET
- Classes used by the application that reference .NET objects
- Third-party .NET controls
Deploying an Application

The **IntelliStream System Tasks Definition** dialog includes functionality to aid you in deploying the assemblies for the run-time versions of the OpenEdge Ultra Controls for .NET:

![IntelliStream System Tasks Definition dialog]

This functionality reads your application’s Assembly References File (assemblies.xml) and chooses only those Ultra Controls used in your application for deployment. For more information on using this functionality, see the online help for the WebClient Application Assembler.

**Note:** Because other applications might require them, uninstalling WebClient or your WebClient application does not uninstall the OpenEdge Ultra Controls for .NET.

You must ensure that you deploy assemblies in the correct location. Progress Software Corporation recommends that you deploy assemblies in the “At Startup” component to ensure that they are available when required. If your application includes third-party assemblies that must be installed in the Global Assembly Cache (GAC), you must include a separate installer for them that the WebClient can launch. For example, you might add an IntelliStream ABL Install Procedure to your WebClient deployment that executes the necessary installers.

For an Administrator installation, the WebClient Initializer defaults to copying the controls to the `<WebClient-install>in` directory. For a personal installation, it defaults to copying the controls to the `<WebClient-App-install>` directory. However, if the application’s command line includes the Assemblies (-assemblies) startup parameter, the Initializer copies the controls to that directory instead for both installation types.

For more information on deploying assemblies, see the chapter on maintaining user environments in *OpenEdge Deployment: Managing ABL Applications*.

**Other files**

In addition to assemblies, every OpenEdge GUI application requires an Assembly References File (assemblies.xml). You must deploy this file in the “At Startup” component to ensure proper execution of your WebClient application.

In exceptional circumstances, your OpenEdge GUI application might use an assemblies.config file. You must deploy this file in the “At Startup” component to ensure proper execution of your WebClient application.

For more information on these files, see the chapter on maintaining user environments in *OpenEdge Deployment: Managing ABL Applications*. 
Your End Users’ Experience

Before they can do anything, end users need installation information and application instructions. After WebClient and the application are installed, they might need information about the WebClient Application Manager, which enables them to view and change information about the WebClient and application installation. These topics are outlined in the following sections:

- Preparing documentation for end users
- WebClient Application Manager
Preparing documentation for end users

You must provide documentation for your users that includes following information:

- You should inform users which installation types you support. If your application installer includes any of the System Tasks that require Administrator privileges, you should specify what they are. If your end users might take the option of granting specific write privileges to non-Administrators, you must detail exactly what privileges are required. For more information on these topics, see the “WebClient installation types” section on page 1–6 and the “Granting write privileges” section on page 6–3.

- If your application uses the OpenEdge GUI for .NET, you should inform users what the appropriate versions of the .NET framework are for running your application. Explain what they need to do, based on the choice you made about including a .NET framework installer. If your WebClient installer launches a .NET Framework installer, tell your end users to accept the EULA and install the framework. For more information, see the “.NET Framework” section on page 6–44.

- The end user’s machine must run a Web browser that supports ActiveX controls or plug-in technology, such as Microsoft Internet Explorer 6 and Firefox 2 (or later versions).

- You should inform users that during WebClient installation they must provide the proxy host and port of any firewall on their system. For more information, see the “WebClient Application Manager” section on page 7–4.

- You should warn users that during the WebClient and application installation they might be told to reboot their machines after the WebClient and application are installed.

- To run your application, users must have Web access. Even though the application is installed on a user’s machine, WebClient Initializer checks the application configuration file on your Web site each time the application is run to verify whether there is a new version of WebClient or the application.

If their browser displays an alert box similar to the one shown, users must choose the Open it option to install or start the application:
Users can clear the **Always ask before opening this type of file** check box if they do not want to see this alert box the next time they launch the application from their browser.

**Note:** Clearing this check box turns off notification for all WebClient applications that the user launches from their browser.

- When users first install WebClient from your Web site, the browser might require them to download a setup player for the installer. When they download the player, a dialog prompts them to grant a certificate that allows the player to be installed. Advise your users that they must always choose to grant the certificate.

- You must tell users how to run the WebClient Application Manager to access the WebClient Initializer log file or make any post-installation changes to their application configuration. For more information see the “WebClient Application Manager” section on page 7–4.

- If you use IntelliStream to install your application, tell your end users that, when uninstalling, they must first uninstall your WebClient application and then uninstall the WebClient itself.

- If your application performs HTTPS connections to a Web server that uses digital certificates signed by a private CA, you must provide your end users with documentation about your installation package for digital certificates. Your documentation should direct the end user to first download and install the WebClient and then immediately install the root digital certificates. Your documentation should also provide your end users with detailed information about how they would update the root digital certificates for the WebClient. Because each public or private PKI vendor has different procedures for installing and using digital certificates, you need to provide explicit directions for each PKI vendor you use. For more information, see Chapter 4, “Designing Security.”
**WebClient Application Manager**

WebClient Application Manager enables end users to configure and manage their use of a WebClient application and to provide information for troubleshooting. End users should only run the manager with your supervision, and usually for the following reasons:

- To specify post-installation WebClient startup parameters
- To specify the proxy server host and port for the user’s firewall
- To access information such as the WebClient Initializer log file and the system registry settings to help you to debug any installation problems

The manager is installed as part of the WebClient installation. Although end users might have multiple versions of WebClient installed on their machine, there is always only one version of the manager for each installation type. You would have a single WebClient Application Manager from all your Administrator instances of WebClient, and another WebClient Application Manager from all your personal instances. The manager for each type of installation is from the most current version of WebClient for that installation type. However, although the manager is part of most current WebClient installation, it can display information about all instances installed on a user’s machine.

**Using the WebClient Application Manager**

The WebClient Application Manager provides end users with a way to view and edit information about the WebClient applications installed on their system. The manager gives access to information about the WebClient instances and instances of WebClient applications. For example:

- **WebClient instances** — Lists the WebClient versions installed on the user’s system and enables the user to modify general options for WebClient execution
- **Applications instances** — Lists the WebClient applications installed on the user’s system by vendor name and application name, and enables the user to review, set, or change execution options for a selected application

The user can start the manager in one of the following modes:

- **WebClient** — This mode is available from a shortcut created by the WebClient installation. The WebClient installer creates this shortcut by default on the Start menu under the WebClient program folder. When this shortcut is used, the manager appears with the **WebClient** tab selected. For more information, see the “WebClient tab” section on page 7–5.
- **Application** — You can make this mode available to the end user through a shortcut that you set up during the application installation. When this shortcut is used, the manager appears with the **Applications** tab and current application selected. For more information, see the “Applications tab” section on page 7–6.

For more information about setting up the shortcut for this mode, see the “External installer” section on page 6–7, and the “System tasks” section on page 6–7.
WebClient tab

If the user runs WebClient Application Manager in WebClient mode, the manager appears with the **WebClient** tab selected. When a user runs the manager, it now shows all Administrator instances of WebClient on the machine, but only that user’s personal instances of WebClient. The manager differentiates between the two installation types by showing personal instances in grey text.

In this example, the **WebClient** tab displays a R10.1C Administrator instance and a R10.2B personal instance:

The **WebClient** tab includes the **System Settings** controls, enabling the end user to view and edit the current settings for the application directory, whether or not the persistent authentication cache is enabled, and the host and port of a proxy server if the user’s system must connect to your Web server through an HTTP-based proxy server (firewall). If the user provided the proxy host and port during WebClient installation, these fields display them as the initial values. If the current user does not have write access for the selected instance, the manager does not enable the values for editing.

The user can modify these settings as follows:

- **Application Dir** — Allows users to change the directory on their system where a WebClient application is installed by default. The users would change this directory when there is not enough space on the drive initially specified by the user for the application installation.

- **Proxy Host** — Allows the user to set or change the name or IP address of the host for the user’s firewall. This option appends the `–proxyhost` startup parameter with the associated host information to the list of startup parameters used to execute WebClient.

- **Proxy Port** — Allows the user to set or change the port for the user’s firewall. This option appends the `–proxyport` startup parameter with the associated port information to the list of startup parameters used to execute WebClient.

- **Persistent Authentication Cache** — Indicates if the proxy server authentication information is cached. You can use the **Clear Cache** button to clear the registry entries for the associated security cache.

**Caution:** Clearing the cache is permanent and cannot be undone.
Applications tab

The Applications tab displays a list of all WebClient application instances currently installed on the user’s system. Applications are identified using the vendor name and the application name. In this example, the Applications tab displays an Administrator instance of the MyApp application and a personal instance of the MyApp2 application:

To make changes to an application’s options, or to view the WebClient application log file, select the application in the list and click Edit.

Edit dialog

The Edit dialog displays information about WebClient applications and allows users to change certain options for the application selected in the Applications tab. The information displayed reflects the information about the last time the application ran. If the current user does not have write access for the instance, the manager does not enable the values for editing.

The Edit dialog title reflects the name of the selected application. The tabs that appear in the Edit dialog vary depending on whether or not you use the WebClient Application Assembler to define the deployment. For all applications, the following tabs appear in the Edit dialog:

- **General** — Lists general information about the selected application, and allows the user to set or change certain execution options. See the “General tab” section on page 7–7.
- **Log** — Displays the contents of the application log file generated by WebClient, and allows the user to delete the contents of the file. See the “Log tab” section on page 7–9.

For application installations that were defined using the WebClient Application Assembler, the following additional tabs appear in the Edit dialog:

- **Security** — Displays security information about the selected application. For more information, see the “Security tab” section on page 7–9.
- **Component** — Displays information about the application components that are installed on the end users system. For more information, see the “Component tab” section on page 7–10.

Clicking Write to File copies all the registry values associated with the selected application to a file you specify. This information is used to help you troubleshoot problems that occur when the application runs.
**General tab**

The **General** tab displays mostly read-only information about the application. However, you can use this tab to have your users add startup parameters for the application. The information displayed reflects the status of the application the last time it was run on this system.

The fields that appear in the **General** tab vary depending on whether or not you use the WebClient Application Assembler to define the deployment. Figure 7–1 and Figure 7–2 show the difference in the information displayed.

![Figure 7–1: General tab with WebClient Application Assembler info](image1)

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vendor Name</td>
<td>Progress</td>
</tr>
<tr>
<td>Configuration File</td>
<td>\<a href="http://hostname/Config.exe">http://hostname/Config.exe</a></td>
</tr>
<tr>
<td>Application Dir</td>
<td>d:\WebClient\Progress</td>
</tr>
<tr>
<td>Local Config File</td>
<td>d:\WebClient\Config.exe</td>
</tr>
<tr>
<td>Application Version</td>
<td>2.0</td>
</tr>
<tr>
<td>System Task Version</td>
<td>1.0</td>
</tr>
<tr>
<td>4CL Install Proc Version</td>
<td>1.0.3</td>
</tr>
<tr>
<td>External Install Version</td>
<td>1.5</td>
</tr>
<tr>
<td>WebClient Version</td>
<td>9.1c</td>
</tr>
<tr>
<td>Additional Startup Parameters</td>
<td>main</td>
</tr>
</tbody>
</table>

![Figure 7–2: General tab without WebClient Application Assembler info](image2)

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vendor Name</td>
<td>Progress</td>
</tr>
<tr>
<td>Program File</td>
<td>\<a href="http://hostname/31b89f06.exe">http://hostname/31b89f06.exe</a></td>
</tr>
<tr>
<td>Application Dir</td>
<td>d:\WebClient\Progress</td>
</tr>
<tr>
<td>Application Version</td>
<td>1.0</td>
</tr>
<tr>
<td>WebClient Version</td>
<td>9.1b</td>
</tr>
<tr>
<td>Program Version</td>
<td>1.0</td>
</tr>
<tr>
<td>Additional Startup Parameters</td>
<td>main</td>
</tr>
</tbody>
</table>

For all applications, the following fields appear in the **General** tab:

- **Vendor name** — Name of the application provider.
- **Application Dir** — Local directory where the application is installed, which the application uses as its execution working directory.
- **Application Version** — Current version of the application.
- **WebClient Version** — Version of WebClient used to run the application.
• **Additional Startup Parameters** — Editable field that allows the user to enter any well-formed list of OpenEdge startup parameters using the syntax you provide to them. For more information on the available startup parameters, see *OpenEdge Deployment: Startup Command and Parameter Reference*.

Your users might use these parameters to improve performance specific to this user. WebClient Initializer appends all startup parameters specified in this field to the end of the list of startup parameters already specified.

If the application installation was not defined with the WebClient Application Assembler, as shown in Figure 7–2, the following additional fields appear in the **General** tab:

• **ProwcApp File** — URL for the configuration file used to install and start the application

• **ProwcApp Version** — Version number used for the benefit of the ISV

If the application installation was defined with the WebClient Application Assembler, as shown in Figure 7–1, the following additional fields appear in the **General** tab:

• **Configuration File** — URL for the configuration file used to install and start the application. This file (-prowcapp) specifies the name of the WebClient configuration file (.prowcapp) used when deploying an application. The WebClient Application Assembler creates and downloads this configuration file to the end user’s machine each time a WebClient application starts. This file provides all the information necessary for the WebClient application. You refer to this file with the following syntax:

```
-prowcapp filename
```

Where *filename* represents the pathname of the application configuration file (.prowcapp).

• **Local Config File** — Pathname of the configuration file cached on the end user’s machine.

• **System Tasks Version** — Current version of system tasks as defined in the WebClient Application Assembler.

• **ABL Install Proc Version** — Current version of the ABL install procedure as defined in the WebClient Application Assembler.

• **External Install Version** — Current version of the external installer as defined in the WebClient Application Assembler.

**Caution:** WebClient Application Manager does not validate the data the user enters.
Log tab

The Log tab displays the contents of the application log file generated by WebClient:

(WebClient writes status messages to a log file for you to troubleshoot problems. The log file is created in the Application Dir you specified.

The path to the log file appears above the file contents box so that the end user can locate the file if you or Progress Technical Support ask for it. The user can delete the application log file by choosing the Clear Log button.

Caution: Clearing the log is immediate and permanent. Choosing the Cancel button does not undo deletion of the application log file.

Security tab

The Security tab displays authentication information about the selected application. This tab appears only when you use the WebClient Application Assembler to define the deployment:
The following fields appear on the **Security** tab:

- **Application Security Information** — Indicates whether or not the application is signed. The possible values are **Signed** and **Not Signed**.

- **Persistent Authentication Cache** — Indicates if the **Config File Cache** and the **Codebase Cache** are being used. The possible values are:
  
  - **Used** — Authentication information is cached
  
  - **Not Used** — Authentication information is not cached

You can use the **Clear Cache** button to clear the registry entries for the associated security cache. This button is not available when the cache is not used.

**Caution:** Clearing the cache is permanent and cannot be undone.

**Component tab**

The **Component** tab provides a list of application components that have been downloaded to the end user’s machine. This tab appears only when you use the WebClient Application Assembler to define the deployment:
Deploying the Sample Application with IntelliStream

This appendix tells you how to deploy the OpenEdge sample application, SportsPro, with WebClient with IntelliStream as described in the following sections:

- Preparing to run the sample application
- Preparing the Web server
- Setting up the application on the AppServer machine
- Setting up the application to launch from the Web server
- Installing and running the application on the client machine

Note: These procedures assume that you are installing as an Administrator.
Preparing to run the sample application

The WebClient files that support running the sample application with IntelliStream are located in directories on the Documentation and Samples located in the doc_samples directory on the OpenEdge Product DVD and the Progress Documentation Web site. For more information on accessing these directories, see the section on example procedures in the “Preface”. Instructions for using these files are located in the following file:

```
Doc_and_Samples_Install\src\samples\webclient\dynappdel\instructions.txt
```

When you run the sample application with IntelliStream, only a portion of the application downloads initially, while the rest downloads only as needed.

**Note:** There is no difference between SportsPro in the current version of OpenEdge and previous versions of Sports2000.

The instructions in this appendix use the following symbols, some of which might be defined as environment variables on your system:

- `%DLC%` (sometimes also indicated as `OpenEdge_install_dir`) refers to the OpenEdge installation directory on a particular machine. For example, the instructions might mention `%DLC%` on the end user-machine.
- `%DOCS%` (sometimes also indicated as `Doc_and_Samples_Install`) refers to the Documentation and Samples located in the `doc_samples` directory on the OpenEdge Product DVD.
- `%ASWORKDIR%` refers to the work directory of an AppServer.
- `%APPROOTDIR%` refers to the root directory of a WebClient application.
- `%OUTDIR%` refers to the directory in which the WebClient application assembler places the files it generates.
- `%MYWEBSERVER%` represents your Web server’s domain name or IP address.
Preparing the Web server

Preparing the Web server involves:

- Setting up the Java Servlet Engine and the AppServer Internet Adapter
- Setting up MIME types for the application configuration file
- Setting up the WebClient install and bootstrap mechanism

**Note:** The application configuration file always has a file extension of .prowcapp and always resides in a Microsoft Cabinet file with a file extension of .prowcapc. This is true whether the application is signed or unsigned.

**Setting up the Java Servlet Engine and the AppServer Internet Adapter**

For instructions on setting up the Java Servlet Engine (JSE) and the AIA on the Web server, see *OpenEdge Application Server: Administration* and the Progress Technical Support Knowledge Base entries on setting up the JSE to work with the AIA. To access these, search the Knowledge Base on “JSE” and “AIA.”

**Setting up MIME types for the application configuration file**

On the Web server, set up MIME types for the application configuration file. The technique for doing so depends on your particular Web server. For information on setting up MIME types for a particular Web server, see the “Configuring MIME types for your Web server” section on page 6–30. Also, see the documentation for your Web server.
Setting up the WebClient install and bootstrap mechanism

The following section provides information about the WebClient and bootstrapping.

**Note:** The sample files on the CD are read-only. You need to edit their properties to make them read-write.

To set up the WebClient install and bootstrap mechanism:

1. Recursively copy the `webinstall\webclient\web_image` directory located in your OpenEdge installation directory to the `web_image` subdirectory of your Web server's document root directory.

2. Edit the `bootstrap.htm` file (which was copied in Step 1) so it references your WebClient install image and your application configuration file. In the `bootstrap.htm` file, replace the `document.location.href` and `window.location` URLs to match the location of your configuration and `webclient.htm` files, respectively. For example:

   ```html
   document.location.href = "http://pctest/web_image/sportspro.prowcapc";
   window.location = "http://pctest/web_image/webclient.htm";
   ```

   **Note:** Do not edit these files in Notepad. Notepad might change the line terminators, which can cause problems with your browser.

3. Edit the `webclient.htm` file (which was copied in Step 1) as follows:

   a. Uncomment the lines that invoke the WebClient initializer. (Keep commented the lines that invoke the One-Click application install directly.)

   b. Change `<URL to your .prowcapc file> to the http://%MYWEBSERVER%/web_image/sportspro.prowcapc URL. For example:

      ```javascript
      ether.SetProperty("ProwcappURL",
      "http://pctest/web_image/sportpro.prowcapc");
      ```
Setting up the application on the AppServer machine

This section provides information about applications on AppServer machines.

**Note:** This example assumes that the AppServer machine is the same as the development machine on which you prepare the front-end and back-end application code. For this reason, this machine must be a Windows machine.

To set up the application on the AppServer machine:

1. Verify that an **OpenEdge development** product is installed on the AppServer machine. The person doing the installation, when presented with the **Setup Type** window, must choose the **complete installation**, as opposed to the typical installation or the custom installation.

2. Verify that an AppServer was installed and configured on the AppServer machine.

3. Create the WebClient application root directory, the symbol for which is `%APPROOTDIR%`. The application root directory can reside anywhere, but note its path.

4. Recursively copy the source-code files of the Sports2000 application from the `%DOCS%\src\sports2000\gui` directory to `%ASWORKDIR%`. **Note:** The `gui` directory contains both back-end and front-end application files. `%ASWORKDIR%` is where the entire application is compiled, though only the back-end r-code (.r) files are run from this directory.

5. Create a copy of the Sports2000 database on the AppServer machine. The copy can reside anywhere, but be sure to note its path.

6. Start an OpenEdge session and do the following:
   
   a. Connect to the Sports2000 database, entering its path, which you noted in Step 5.
   
   b. Use the PROPATH Editor to add the AppServer work directory to the PROPATH.
   
   c. In the Application Compiler, go to **Options → Compiler**, change **Minimize R-code size** to **YES**, change **Generate MD-5** to **YES**, and click **OK**. In the main window, click **Modify**. In the **File Specification** window, enter the AppServer’s working directory, click **OK**, and compile the application.

7. Recursively copy the directories in the following table as specified, creating the destination subdirectory as necessary:

<table>
<thead>
<tr>
<th>Recursively copy directory . . .</th>
<th>To location . . .</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>%ASWORKDIR%\gui</code></td>
<td><code>%APPROOTDIR%\app1</code></td>
</tr>
<tr>
<td><code>%DLC%\gui\adm2</code></td>
<td><code>%APPROOTDIR%\gui\adm2</code></td>
</tr>
<tr>
<td><code>%DLC%\jms</code></td>
<td><code>%APPROOTDIR%\jms</code></td>
</tr>
<tr>
<td><code>%DOCS%\src\sports2000\images</code></td>
<td><code>%APPROOTDIR%\app1\images</code></td>
</tr>
</tbody>
</table>
8. Copy the files in the following table as specified, creating the destination subdirectory as necessary:

<table>
<thead>
<tr>
<th>Copy file</th>
<th>To location</th>
</tr>
</thead>
<tbody>
<tr>
<td>%DOCS%\src\samples\webclient\dynappdel\sportspro.ini</td>
<td>%APPROOTDIR%</td>
</tr>
<tr>
<td>%DOCS%\src\samples\webclient\dynappdel\appsrvt.d</td>
<td>%APPROOTDIR%</td>
</tr>
<tr>
<td>%DOCS%\src\samples\webclient\dynappdel\sportspro.wcp</td>
<td>%APPROOTDIR%</td>
</tr>
<tr>
<td>%DLC%\gui\adecomm.pl</td>
<td>%APPROOTDIR%\gui</td>
</tr>
<tr>
<td>%DLC%\gui\adeicon.pl</td>
<td>%APPROOTDIR%\gui</td>
</tr>
<tr>
<td>%DLC%\gui\adexml.pl</td>
<td>%APPROOTDIR%\gui</td>
</tr>
<tr>
<td>%DLC%\gui\adeicon\progress.ico</td>
<td>%APPROOTDIR%</td>
</tr>
<tr>
<td>%DLC%\bin\system\csspin32.ocx</td>
<td>%APPROOTDIR%</td>
</tr>
</tbody>
</table>

9. Use a text editor to modify the appsrvt.d file, which you copied in Step 8, as follows:
   a. In appsrvt.d, search for the following string:


   b. In the string, change yourservername to the domain name of the Web server running the AIA. For example, if the domain name of the Web server running the AIA is http://testhost, modify the string to appear as follows:


   c. Save appsrvt.d.

10. Use the WebClient Application Assembler as follows:
   a. From the main menu, select **File**→**Open**, and open %APPROOTDIR%\sportspro.wcp.
   b. On the **General** tab, change the path of the application root directory to its actual path on the AppServer machine. The default value is d:\demo91c\sportspro.
   c. On the same tab, click on the **Locator** button. The **Configuration File Locator Definition** window appears. In this window, change the URL to the host name of the server hosting the configuration file, which defaults to http://yourhostname.
   d. In the **Codebase Locator Definition** window (main menu→**Deployment**→**Locators**→**Codebase**), change the URL to the host name of the server hosting the codebase, which defaults to http://yourhostname.
e. On the **Options** tab, check the **System Tasks** toggle box and click **System Tasks Definition**. The **System Tasks Definition** window appears.

On the **Shortcut** tab, check the **Application Shortcut** toggle box and click the ellipsis (…) button next to the **Icon** field. The **Open** window appears.

Manipulate the **Open** window to select the progress.ico file, which you copied to %APPROOTDIR% in Step 7.

On the **Files** tab, select the file d:\demo91c\sportspro\csspin32.ocx and click **Remove**. Then, add the file %APPROOTDIR\csspin32.ocx. Finally, confirm the **Application Specific** toggle box is checked.

f. Save sportspro.wcp.

g. In the **Generate** window (main menu → Deployment → Generate), then choose **Regenerate Latest Version**. In the **Output Directory** field, enter the name of the directory where you want the generated files to be placed; that is, the value of %OUTDIR%. Finally, click **OK**.

The WebClient Application Assembler creates the application configuration file and the cabinet files and places them into the output directory you specified.

11. Use the proserve command or Progress Explorer to start a database server for the sports2000 database running on the AppServer machine. For example, if the sportspro2000 database resides at %APPROOTDIR%\databases, use a command like the following:

```
%DLC%\bin\proserve %APPROOTDIR%\databases\sports2000.db
```

12. Use Progress Explorer to configure the default asbroker1 AppServer instance as follows:

a. In the **General Options for Broker** tab, change the broker working directory to the AppServer work directory (%ASWORKDIR%), and add the service name sports2000 partition to the **Appservice Name** list.

b. In the **General Options for Server** tab, add the database connection parameter to **Server Startup Parameters**. For example if the sports2000 database resides at %ASWORKDIR%\databases, enter the following string:

```
-db %ASWORKDIR%\databases\sports2000
```

13. Use Progress Explorer or the asbman command to start the asbroker1 AppServer instance. For example (using the asbman command):

```
%DLC%\bin\asbman -name asbroker1 -start
```
Setting up the application to launch from the Web server

This section provides information about applications and Web servers.

To set up the application to launch from your Web server machine:

1. Copy `sportspro.prowcapc` (the cabinet file containing the configuration file to be downloaded) and the cabinet (.cab) files (containing the application files to be downloaded) from `%OUTDIR%` on the AppServer machine to the Web server root directory as follows:
   a. Create a directory called `2.0` off the Web server document root directory.
   b. Copy the cabinet files to the `2.0` directory.
   c. Copy `sportspro.prowcapc` to the Web server document root directory.

2. Configure (if not already configured) and start an instance of the AIA to run as a Java servlet in the JSE and to access the AppServer served by `asbroker1`. 
Installing and running the application on the client machine

This section provides information about applications on client machines.

To install and launch the application on the client machine:

1. Use an appropriate Web browser to go to the following URL, where YourDomainName represents the domain name of your Web server:

   http://YourDomainName/web_image/bootstrap.htm

2. To install WebClient and the Sportspro application, respond to the prompts. The application is installed.

Note: The sample application’s SmartB2B and SonicMQ buttons invoke options not yet implemented.
Deploying the Sample Application
Without IntelliStream

This appendix tells you how to deploy the OpenEdge sample application, SportsPro, with WebClient without IntelliStream, as outlined in the following sections:

• Preparing to run the sample application
• Preparing the Web server
• Setting up the application on the AppServer machine
• Setting up the application on the Web server machine
• Installing and running the application on the client machine

Note: These procedures assume that you are installing as an Administrator.
Preparing to run the sample application

Some files required to deploy the sample application without IntelliStream are located in the following directories on the Documentation and Samples located in the \doc\samples\ directory on the OpenEdge Product DVD and the Progress Documentation Web site:

```
$Doc_and_Samples_Install\webinstall\$n
$Doc_and_Samples_Install\src\sports2000$
```

For more information on accessing these directories, see the “Example procedures” section on page Preface–5.

The instructions in this appendix use the following symbols, some of which might be defined as environment variables on your system:

- `%DLC%` (sometimes also indicated as `OpenEdge_install_dir`) represents the OpenEdge installation directory on a particular machine. For example, the instructions might mention `%DLC%` on the end-user machine.

- `%DOCS%` (sometimes also indicated as `Doc_and_Samples_Install`) refers to the Documentation and Samples located in the `doc\samples\` directory on the OpenEdge Product DVD.

- `%ASWORKDIR%` represents an AppServer’s work directory.

- `%MYWEBSERVER%` represents your Web server’s domain name or IP address.

**Note:** There is no difference between the SportsPro application in the current version of OpenEdge and the previous versions of Sports2000 applications.
Preparing the Web server

Preparing the Web server involves two steps:

- Setting up the Java Servlet Engine and the AppServer Internet Adapter
- Setting up MIME types for the application configuration file

Setting up the Java Servlet Engine and the AppServer Internet Adapter

For instructions on setting up the Java Servlet Engine (JSE) and the AppServer Internet Adapter (AIA) on the Web server, see *OpenEdge Application Server: Administration* and the Progress Technical Support Knowledge Base entries on setting up the JSE to work with the AIA. To access these, search the Knowledge Base on “JSE” and “AIA.”

Setting up MIME types for the application configuration file

On the Web server, set up MIME types for the application configuration file. The technique for doing so depends on your particular Web server. For information on setting up MIME types for a particular Web server, see the “Configuring MIME types for your Web server” section on page 7–27. Also, see the documentation for your Web server.
Setting up the application on the AppServer machine

This section provides information about applications on AppServer machines.

Note: The sample files on the CD are read-only. You need to edit their properties to make them read-write.

To set up the application on the AppServer machine:

1. Verify that a complete installation of an OpenEdge development product was installed on the AppServer machine.

2. Recursively copy the back-end files of the Sports2000 application from the %DOCS%\src\sports2000\gui directory to the AppServer work directory.

   Note: The gui directory contains both back-end and front-end application files.

3. Create a copy of the sports2000 database on the AppServer machine. The copy can reside anywhere, but note its full path. For example, d:\wrk91b\sports2000.db.

4. Start a database server for the sports2000 database running on the AppServer machine. Use Progress Explorer or a proserve command similar to the following:

   %DLC%\bin\proserve d:\wrk91b\sports2000.db

5. Compile the sports2000 back-end application files you copied to the AppServer working directory in Step 2, as follows:

   a. Start an OpenEdge session.

   b. Connect to the sports2000 database.

   c. Use the PRO*Tools PROPATH tool to add the AppServer working directory to the PROPATH.

   d. Start the Application Compiler.

   e. Enter the path of the AppServer working directory and compile the application.
6. Use Progress Explorer to configure the default asbroker1 AppServer instance, as follows:

   a. In the General Options for Broker tab, change the broker working directory to the AppServer work directory, and add the service name sports2000partition to the Appservice Name list.

   b. In the General Options for Server tab, add the database connection parameter to Server Startup Parameters. For example, if the Sports2000 database resides at d:\wrk91b, add the following parameter:

   ```
   -db d:\wrk91b\sports2000
   ```

7. Use Progress Explorer to configure and start the controlling NameServer instance for the asbroker1 AppServer.

8. Use Progress Explorer or the asbman command to start the asbroker1 AppServer instance. For example (using the asbman command):

   ```
   %DLC%\bin\asbman -name asbroker1 -start
   ```
Setting up the application on the Web server machine

This section provides information about applications on Web server machines.

To set up the application to launch from your Web server machine:

1. Recursively copy all files from the %DOCS%\webinstall\sports2000\web_image directory to the sports2000 subdirectory of the Web server document root directory.

2. Copy all files from your %DLCS%\webinstall\webclient\web_image directory to the Web server document root directory’s webclient subdirectory.

3. In the Web server document root directory’s webclient subdirectory, edit the webclient.htm file to install the application automatically just after installing WebClient. To do this:
   a. Uncomment the lines that invoke the One-Click application install directly. (Keep commented the lines that invoke the WebClient initializer.)
   b. Change <URL to your One-Click application installer directory> to the http://%MYWEBSERVER%/sports2000 URL.
   c. Change <URL to your .prowcapc file> to the http://%MYWEBSERVER%/sports2000/sports2000.prowcapc URL. For example:

   ```
   ```

   These changes let you install WebClient and the Sports2000 application on the client machine.

4. In the Web server document root directory’s webclient subdirectory, make the following changes:
   a. Edit the bootstrap.htm file, replacing the document.location.href and window.location URLs to match the location of your configuration and webclient.htm files, respectively. For example:

   ```
   window.location = "http://pctest/webclient/webclient.htm";
   ```

Note: Do not edit these files in Notepad. Notepad might change the line terminators, which can cause problems with your browser.
b. Edit the sports2000.htm file, uncommenting the line containing the prowcapc entry, and including the domain name or IP address of your Web server and the name of your application configuration file. For example:

```java
```

5. Use the WebClient Application Assembler PRO*Tool to generate a customized sports2000.prowcapp file. To do this:
   a. Copy %DOCS%\webinstall\sports2000\sports2000.wcp to any directory.
   b. Open the file with the WebClient Application Assembler.
   c. Modify the URL of your configuration file specified in the **Configuration File Locator** window, the URL of the WebClient install on the **WebClient** tab, and the URL for your application install specified on the **Options** tab.
   d. From the WebClient Application Assembler’s main menu, choose **Deployment → Generate**. Enter an output directory and a version name (for example, v1), and click **OK** to generate a new configuration file.
   e. Copy the configuration file to the Web server document root directory’s sports2000 subdirectory.

6. Configure (if not already configured) and start an instance of the AppServer Internet Adapter to run as a Java servlet in the Java Servlet Engine and to access the AppServer served by asbroker1.

For instructions on setting up the Java Servlet Engine (JSE) and the AppServer Internet Adapter (AIA) on the Web server, see *OpenEdge Application Server: Administration* and the Progress Technical Support Knowledge Base entries on setting up the JSE to work with the AIA. To access these, search the Knowledge Base on “JSE” and “AIA.”
Installing and running the application on the client machine

This section provides information about applications on client machines.

To install and launch the application on the client machine:

**Note:** Normally Step 1 would not be necessary. But since it was impossible to predict the name of your server, the `appservtt.t` file, which contains the connection URL and is included in the installation materials, has been set to `YourServername`.

1. Edit the hosts (`winnt\system32\drivers\etc\hosts`) file on the client machine to add an entry containing the server’s IP address and host name. The entry might appear as follows:

   | 173.16.114.111 | `YourServername` |

2. Open an appropriate Web browser and go to one of the following URLs:

   - `http://%MYWEBSERVER%/webclient/bootstrap.htm`
   - `http://%MYWEBSERVER%/sports2000/sports2000.prowcapc`

   For example:

3. Install WebClient and the Sports2000 application by following the instructions and accepting the defaults.

**Note:** The sample application’s SmartB2B and SonicMQ buttons invoke options not yet implemented.
# Index

## A

- ABL for Web-enablement
  - authentication 5–13
  - LOAD–IMAGE method 5–11
  - PROPATH 5–11
  - SEARCH function 5–11
- ABL install procedure 6–7, 6–18
  - WebClient applications 2–4
- Administrator installations 1–6, 6–3
  - granting privileges
    - Registry keys
      - granting privileges 6–3
- Administrator users 1–7
- Application configuration file 1–4
- Applications
  - compiling 5–14
  - component files 6–22
  - components 6–6
  - components update files 6–22
  - configuration file 6–21
  - mode 7–4
  - registry keys 6–9, 6–11
  - signed and unsigned 6–25
- As Needed components 1–13
- Ask at Startup components 1–13
- ASYNC requests 5–6

## Authentication
- dialogs 4–8
  - IIS Web server 6–30
- Authenticity (of a downloaded file) 4–2

## B

- Best practices
  - recommended for WebClient 1–18
- Bootstrap file
  - customizing 6–39
  - starting WebClient applications 6–37
- Bootstrap mechanism A–4
- Bootstrapping
  - with no WebClient installed 6–39
  - with version of WebClient installed 6–39

## C

- Case sensitivity
  - UNIX Web servers 6–29
- CD-ROM
  - installing WebClient 2–2
  - installing WebClient applications 2–4
- Codebase locator 1–12
- CODEBASE-LOCATOR handle 5–2
- Compiling applications 5–14
Components
  application 6–6
  download mode 1–13
  registry keys 6–12
Configuration file locator 1–16
Customizing
  bootstrap.htm 6–39
  WebClient install 2–2
  WebClient installer 6–33, 6–34

D
Deploying WebClient applications
  authentication on Microsoft IIS 6–30
  case sensitivity
    installation filenames 6–29
  end user documentation 7–2
  executable directories 6–30
Deployment configuration
  modifying 6–23 to 6–27
Digital signatures 1–14
Documentation
  for end users 7–2
Download mode
  for components 1–13

E
End user
  providing documentation 2–10
Executable directories
  Web installer 6–30
External installer
  installation overview 2–6
  installation tasks 2–6
  registry keys for uninstall 6–13
  uninstalling WebClient 2–8
  WebClient application updates 2–7
  WebClient applications 2–4
  Web-enabled installer 6–18

F
Features
  IntelliStream 2–5
Firewalls
  setting after WebClient install 7–5

I
IIS Web server
  authentication 6–30
Input files
  WebClient Application Assembler 6–2
Installation types 6–3
  Administrator 1–6, 6–3
  checking 5–5
  choosing 1–19
  granting privileges 6–3
  personal 1–6, 6–4
  precedence 1–7
Installing WebClient applications
  external installer 2–6
  IntelliStream 2–5
  recommendations 2–4
InstallShield One-Click Install
  customizing 6–34
InstallShield Setup Player
  Internet Explorer add-on 6–5
Integrity (of a downloaded file) 4–2
IntelliStream
  application components 6–6
  features 2–5
  installation overview 2–5
  installing WebClient applications 2–4
  uninstalling WebClient 2–8
  WebClient application updates 2–7

K
KEEP-CONNECTION-OPEN attribute 5–2

L
Launching applications
  shortcuts 6–36
  URL 6–36
LOAD-ICON method support for URLs 5–12
LOAD-IMAGE method support for URLs 5–11, 5–12
LOAD-IMAGE support for URLs
  authentication 5–11
LOAD-IMAGE() method 5–11
LOAD-IMAGE-DOWN method support
  for URLs 5–12
LOAD–IMAGE–INSENSITIVE method support for URLs 5–12
LOAD–IMAGE–UP method support for URLs 5–12

M
MD5 message digest 5–14
Microsoft Authenticode Technology 1–4
Microsoft cabinet (.cab) files 1–4
MicroSoft Vista
  WebClient installation considerations 2–2
Microsoft Vista
  User Access Control 2–2
MIME type settings
  Apache Web Server 6–31
  configuring for Web servers 6–30
  Microsoft IIS 6–32
  Sun Java System Web Server 6–31
Modes
  WebClient Application Manager 7–4
Modifying deployment configuration 6–23 to 6–27

N
.NET
  deploying framework installers 6–44
  WebClient considerations 6–44

O
One-Click Install image
  WebClient 2–2
OpenEdge GUI for .NET
  WebClient deployment 6–44
Overview
  uninstalling WebClient 2–8
  WebClient Application Assembler 1–11
  WebClient application updates 2–7
  WebClient install 2–2

P
Personal installations 1–6, 6–3, 6–4
  System Tasks 6–19
Procedure libraries
  updating 6–43
Project file, WebClient (.wcp file) 6–14
PROPATH support for URLs 5–11
  authentication 5–11
Providing documentation
  for end users 2–10
  .prowcapc file 1–12, 6–10, 6–21
    digitally signed 6–25
    modifying 6–23 to 6–27
  .prowcapp file 1–11, 1–12, 6–23
    digitally signed 6–21
    modifying existing 6–23
ProwcappLocator
  registry keys 6–10
Proxy
  –proxyhost startup parameter 7–5
  –proxyport startup parameter 7–5
  servers, setting after WebClient install 7–5

R
Registry keys
  Administrator installs 1–6
  application name 6–9, 6–11
  applications 6–11
  checking programmatically 5–5
  components 6–12
  external installer 6–13
  personal installs 1–6
  ProwcappLocator 6–10
  vendor name 6–9
  WebClient 6–7
  WebClient installer 6–7
Running applications
  Web access 7–2

S
Sample application A–1, B–1
SEARCH function support for URLs 5–11
  authentication 5–11
Security
   designing 4–1 to 4–9
   for custom deployment configuration 6–25

SESSION system handle
   WC-ADMIN-APP attribute 5–5

Setting MIME type
   Apache Web Server 6–31
   Microsoft IIS 6–32
   Sun Java System Web Server 6–31

Shortcuts
   launching applications 6–36
   starting WebClient applications 2–9, 6–36

Single sign-on 1–16, 4–7, 5–7

Source code versioning 5–14

Sports2000 application A–2, B–2

SportsPro application A–1, B–1

Starting WebClient applications 2–9, 6–36

Startup parameters
   proxy host (–proxyhost) 7–5
   proxy port (–proxyport) 7–5

System Tasks
   .NET 6–44
   personal install limits 6–19

System tasks 1–14, 1–16, 6–7
   defining 6–17

T

TEMP–TABLE parameters 1–18

Third-party installers
   assemblies 6–46
   .NET Framework 6–45

U

Uninstalling WebClient
   external installer 2–8
   IntelliStream 2–8
   overview 2–8

Uninstalling WebClient applications
   registry keys 6–13

UNIX Web servers
   case sensitivity 6–29

Updates
   procedure libraries 6–43

URL support
   authentication 5–13
   launching applications 6–36
   LOAD–IMAGE method 5–11
   PROPATH 5–11
   SEARCH function 5–11
   starting WebClient applications 2–9, 6–36

User Access Control (UAC)
   WebClient installs 2–2

V

Vendor name
   registry keys 6–9

Versioning
   source code 5–14

W

WC-ADMIN-APP attribute 5–5

.wcd file 6–23

.wcp file 6–14

Web access
   running applications 7–2
   WebClient applications 7–2

Web authentication
   4GL support 5–13

Web install
   WebClient applications 2–4

Web installer
   executable directories 6–30

Web servers
   configuring MIME types 6–30

WebClient 1–2
   CD-ROM install 2–2
   compared to 4GL client 1–2
   end-user requirements 1–10
   One-Click install image 2–2
   project (.wcp) file 6–14
   registry keys 6–7
   use scenarios 1–3

WebClient Application Assembler 1–11
   generating files 6–6
   input files 6–2
   managing applications 6–1
   overview 1–11
WebClient Application Manager 1–4
Applications tab 7–6
defined 7–4
edit dialog 7–6
functions 7–4
General tab 7–7
Log tab 7–9
modes of execution 7–4
setting a firewall 7–5
shortcut syntax 6–9
WebClient tab 7–5

WebClient application updates
external installer 2–7
IntelliStream 2–7
overview 2–7

WebClient applications
ABL install procedure 2–4
best practices 1–18
CD-ROM install 2–4
external installer 2–4
installation recommendations 2–4
installing 2–4
installing with IntelliStream 2–4
launch mechanism overview 6–36
launch options 6–36
OpenEdge GUI for .NET 6–44
starting 2–9, 6–36
starting with bootstrap file 6–37
URL for starting 6–36
Web install 2–4

WebClient Deployment Packager 6–23 to 6–27
overview 1–12
standalone version 6–23

WebClient deployments
.NET Framework installers 6–44
WebClient executable 1–4

WebClient Initializer 1–2, 1–4
defined 6–36
shortcut syntax 6–8
validating acceptable versions
application 6–41
Progress WebClient 6–41

WebClient install
customizing 2–2
overview 2–2

WebClient installations
in Vista 2–2

WebClient installer
customizing 6–34
registry keys created 6–7
running application installer example 6–35
running WebClient Initializer example 6–35

WebClient mode 7–4

WebClient project (.wcp) file 1–4
webclient.htm 6–34
customizations 6–34
customizing a WebClient installer 6–34

Web-enabled installer
external installer 6–18