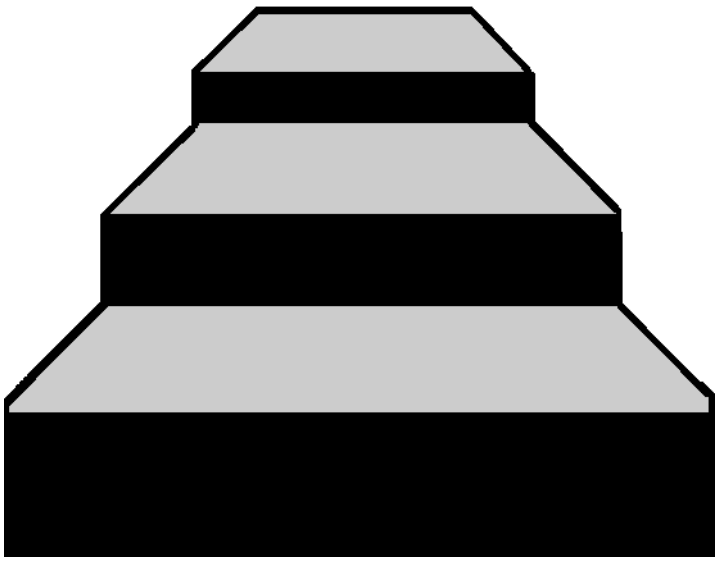
# ST-Developer™ Release Notes



## Version 12

STEP Tools® Software

## Contributors

Written by Jochen Fritz and David Loffredo.

© **Copyright 1991-2007 STEP Tools, Inc. — All Rights Reserved.**

This document contains proprietary and confidential information of STEP Tools, Inc. The contents of this document may not be disclosed to third parties, copied, or duplicated in any form, in whole or in part, without the prior written permission of STEP Tools, Inc.

ST-Developer, ST-Machine, ST-Parasolid, ST-Viewer, and the ST prefix are trademarks of STEP Tools, Inc. STEP Tools is a registered trademark of STEP Tools, Inc. Other brand or product names are the trademarks or registered trademarks of their respective holders.

**STEP Tools, Inc.**
**14 First Street**
**Troy, New York 12180**

**Phone . . . . . (518) 687-2848**
**Fax . . . . . (518) 687-4420**
**E-Mail . . . . . info@steptools.com**
**Web. . . . . http://www.steptools.com**

# Contents

# 1 Release Notes

## 1.1 ST-Developer Version 12

For sixteen years, ST-Developer has been the most complete and mature SDK for building STEP and EXPRESS applications. Our latest release, ST-Developer v12 continues this tradition by adding support for STEP Part 28 XML files, out-of-the-box support for many new STEP Application Protocols, as well as the CIMsteel Integration Standard (CIS/2) and Industry Foundation Classes (IFC), new Visual Studio 2005 support, and many other refinements.

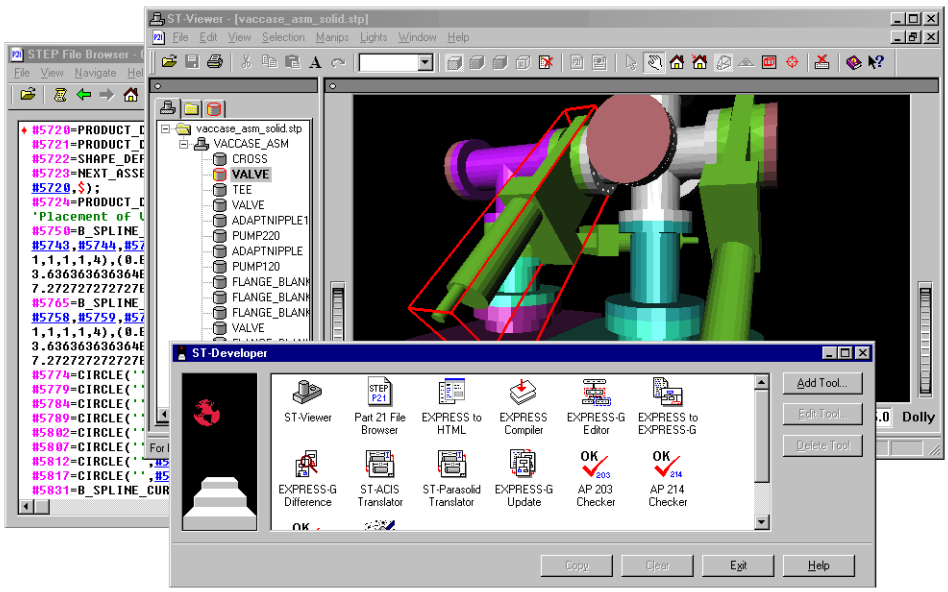



**Figure 1.1 —** ST-Developer Tools

### What's new in ST-Developer v12?

- ST-Developer C++ applications can now read and write STEP Part 28 XML exchange files (ISO 10303-28:2007) in addition to STEP Part 21 text files.
- New support for Visual Studio 2005 with VC++ 2005 compatible libraries and sample projects for all supported APs, as well as continued support for the older Visual C++ 6.0 compiler.
- New platforms for 64bit Windows, MacOS X Intel 32bit, MacOS X Intel 64bit for the upcoming Leopard (10.5), and 64bit Linux, as well as continued support for 32bit Windows, MacOS X PowerPC, 32bit Linux, SPARC Solaris, HP-UX, and SGI Irix.
- New or updated programming libraries for AP203 edition 2, AP224 edition 3, AP219, AP221, AP236, AP238, AP239, AP240, and IFC 2x3 have been added to our continued support for AP201, AP202, AP203, AP209, AP210, AP214, AP215, AP216, AP218, AP225, AP227 AP232, CIS/2, and IFC2x2.
- New ST-Developer Message Window extension to simplify the display of status and debugging information from Windows GUI applications.
- Enhanced C++ support for the EXPRESS "binary" primitive type used by the new AP221 and IFC 2x3 schemas.

ST-Developer continues to be the clear choice for building applications for STEP, STEP-NC, CIS/2, or IFC simply, distributing them easily, viewing, and verifying your data sets.

## 1.2 Supported Platforms

ST-Developer is available for the operating systems and compilers below. Contact STEP Tools to arrange support for other configurations.

Some ST-Developer platforms contain several versions of the programming libraries for use with different C++ compilers or build flags. See **Windows Installation** (Chapter 4, pp. 23), **MacOS Installation** (Chapter 5, pp. 27), and **UNIX Installation** (Chapter 6, pp. 31) for details on each.

| Operating System | Architecture | Supported C++ Compilers |
|---|---|---|
| Windows 2000/XP/Vista | Intel x86 | Visual C++ 8.0 (Visual Studio 2005) and Visual C++ 6.0 with the /MT and /MD options. |
| Windows 64bit | Intel x64<br>AMD64<br>EM64T | Visual C++ 8.0 (Visual Studio 2005) with the /MT and /MD options |
| MacOS X 10.4<br>on PowerPC (Tiger) | PowerPC | GCC 3.3 and GCC 4.0<br>(Xcode 2.0) |
| MacOS X 10.4<br>on Intel (Tiger) | Intel x86 | GCC 4.0 (Xcode 2.3) |
| MacOS X 10.5 64bit<br>on Intel (Leopard) | Intel 64bit | GCC 4.0 (Xcode 2.3) |
| Linux<br>LSB 2.x or 3.x distros, RedHat 9, RHEL 3/4/5, SuSE 9.x and up, and others as described in **Linux Notes** (Section 6.1.1, pp. 31). | Intel x86 | GCC 3.2/3.3 and GCC 3.4/4.x |
| Linux 64bit<br>LSB 3.x distros. | Intel 64bit<br>AMD64<br>EM64T | GCC 3.4/4.x |
| Hewlett Packard<br>HP-UX v11 | PA-RISC | HP ANSI C++ |
| Hewlett Packard<br>HP-UX v11 | Itanium | HP ANSI C++ |
| Silicon Graphics<br>IRIX v6.x | MIPS | SGI C++ default and -n32 |
| Sun Solaris v7 or better | SPARC | Forte 6, Sun Studio v7-11 (with and w/o -PIC -mt), GCC 3.4/4.x |
| Sun Solaris v10 | Intel x86 | Sun Studio v11 64bit default and with -PIC -mt |

## 1.3 Supported Application Protocols

ST-Developer v12 ships with a wide selection of C++ and Java class libraries, HTML-documented schemas, and usage guides. In this release we have added or updated support for AP203 edition 2, AP224 edition 3, AP219, AP221, AP236, AP238, AP239, AP240, and IFC 2x3.

You can start programming immediately, with out-of-the-box support for nineteen STEP Application Protocols, the CIMsteel Integration Standard (CIS/2) as well as two versions of the Industry Foundation Classes (IFC). If we missed your favorite, you can use EXPRESS compiler and other ST-Developer tools to install them yourself! The electronic manuals have quick links to each of the following:

- **AP201: Explicit Draughting** - Includes the schema for ISO 10303-201:1994, plus C++ and Java class libraries, and a sample geometry program.
- **AP202: Associative Draughting** - Includes the schema for ISO 10303-202: 1996, plus C++ and Java class libraries, and a sample geometry program.
- **AP203: Configuration Controlled Design** - Includes the schema for ISO 10303-203:1994/Amd1:2000 and recommended practices, plus C++ and Java class libraries, and five sample programs for geometry, assemblies, and config-control information.
- **AP203 Edition 2: Configuration Controlled Design** - Includes the schema for the technical specification ISO/TS 10303-203:2005, plus C++ and Java class libraries, and four sample programs for geometry, assemblies, and config-control information.
- **AP209: Structural Analysis** - Includes the schema for ISO 10303-209:2001, plus C++ and Java class libraries, recommended practices guide, and a sample geometry program.
- **AP210: Electronic Assembly Interconnect and Packaging Design** - Includes the schema for draft #3 for the TC1 document.
- **AP214: Automotive Design** - Includes the schema for the second edition, ISO 10303-214:2003, plus C++ and Java class libraries, and three sample programs for geometry and assemblies.
- **AP215: Ship Arrangement** - Includes the schema for ISO 10303-215:2004, plus C++ and Java class libraries, and a sample geometry program.
- **AP216: Ship Moulded Form** - Includes the schema for ISO 10303-216:2003, plus C++ and Java class libraries, and a sample geometry program.

- **AP218: Ship Structures** - Includes the schema for ISO 10303-218:2004, plus C++ and Java class libraries, and a sample geometry program.

- **AP219: Dimensional Inspection Information Exchange** - Includes the schema for ISO 10303-219:2007, plus C++ and Java class libraries, and sample programs for geometry and properties, plus a new sample program for creating round hole features.

- **AP221: Functional Data and Schematics for Process Plant -** Includes the MIM long form schema for ISO 10303-221:2007, plus C++ and Java class libraries, and a sample program to create a simple STEP product.

- **AP224: Feature-Based Process Planning** - Includes the schema for the third edition, ISO 10303-224:2006, plus C++ and Java class libraries, and sample programs for geometry and properties, plus a new sample program for creating round hole features.

- **AP225: Building Elements** - Includes the schema for ISO 10303-225:1999, plus C++ and Java class libraries, and a sample geometry program.

- **AP227: Plant Spatial Configuration** - Includes the schema for the second edition, ISO 10303-227:2005, plus C++ and Java class libraries, a usage guide for ship piping, and a sample geometry program.

- **AP232: Technical Data Packaging** - Includes the schema for ISO 10303-232:2002, plus C++ and Java class libraries, the recommended practices guide, and a sample geometry program.

- **AP236: Furniture Catalog and Interior Design** - Includes the MIM long form schema for ISO 10303-236:2006, plus C++ and Java class libraries, and a sample geometry program.

- **AP238: STEP-NC Integrated CNC** - Includes the schema for the STEP-NC ISO 10303-238:2007 standard, plus C++ and Java class libraries, a sample geometry program and a new program to create round hole features, links to web resources, working sets for the various conformance classes.

- **AP239: Product Life Cycle Support** - Includes the MIM long form schema for ISO 10303-239:2005, plus C++ and Java class libraries, and a sample program to create a simple STEP product.

- **AP240: Macro Process Planning** - Includes the schema for ISO 10303-240:2005, plus C++ and Java class libraries, and sample programs to create a simple STEP product with geometry and round hole features.

- **CIMsteel Integration Standard (CIS/2)** - Includes the structural frame schema in the latest LPM/6 version, plus C++ and Java class libraries, a sample ge-

ometry program, and links to web resources. Plus, we have updated the data dictionary with additional complex instances.

- **Industry Foundation Classes Edition 2 (IFC 2x2)** - Includes the schema for the IFC 2x2 Addendum 1 schema (ifc2x2_final), plus C++ and Java class libraries, and a sample geometry program.

- **Industry Foundation Classes Edition 3 (IFC 2x3)** - Includes the schema for the latest IFC 2x3 schema, plus C++ and Java class libraries, and a sample geometry program.

- **STEP Integrated Resources** - Includes browsable EXPRESS for the integrated resources (40-series), application integrated resources (100-series), and the application interpreted constructs (AICs, the 500-series).

- **STEP PDM Schema** - Includes browsable EXPRESS for v1.2 of the STEP PDM schema along with the moste recent recommended practices guide.

### 1.3.1 Note for Upgrading IFC Users

This ST-Developer release contains libraries for both IFC 2x2 and IFC 2x3, so the C++ include path, class libraries and Java jar files are named **ifc2x2lib** and **ifc2x3lib**. The previous release only had libraries for IFC 2x2 with resources named **ifclib**.

When you move to the latest release, you will need to change your project settings to refer to **ifc2x2lib** or **ifc2x3lib** rather than **ifclib.**

## 1.4 Part 28 XML Support

ST-Developer C++ applications can now read and write STEP Part 28 Edition 2 XML exchange files (ISO 10303-28:2007). All of the sample programs have been extended to include this support. To enable this functionality in your own application, simply link against the **p28e2** library, include the **rose_p28.h** file, then call **rose_p28_init()** at the start of your program to register the XML reader and writer.

From then on, your application will automatically recognize STEP Part 28 XML files when reading. By default, new files are written as Part 21 but you can switch to Part 28 XML by calling **RoseDesign::format()** with "**p28**" or **p28-raw**" strings.

```
RoseDesign * d;
d-> format ("p21");
```

```
d-> save()  // write as Part 21 text

d-> format ("p28");
d-> save()  // write compressed Part 28 XML

d-> format ("p28-raw");
d-> save()  // write raw Part 28 XML
```

When the "**p28**" format is specified, the XML will be zip compressed to reduce size, and the file will be given the "**.p28**" extension if none is specified. If you run unzip on the result, you will see a single file called **iso_10303_28.xml**, which contains the raw XML.

With the "**p28-raw**" format is specified, the XML will be written without any compression, and the file will be given the "**.xml**" extension if none is specified.

The small Part 28 XML file below shows some AP238 instance data with an axis placement, cartesian point, and two directions. The specifics of the Part 28 encoding are discussed in Chapter 2 of the *ROSE Library Reference Manual*

```
<?xml version="1.0"?>
<iso_10303_28_terse
    xmlns="urn:oid:1.0.10303.238.1.0.1"
    xmlns:exp="urn:oid:1.0.10303.28.2.1.1"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    schema="integrated_cnc_schema">

   <exp:header>
     <exp:name>dataset</exp:name>
     <exp:preprocessor_version>stepnc writer</exp:preprocessor_version>
   </exp:header>

  <!-- instance data -->
  <Axis2_placement_3d id="id1" Name="" Location="id2" Axis="id3"
    Ref_direction="id4"/>
  <Cartesian_point id="id2" Name="loc" Coordinates="3.5 -3.5 -4.16875"/>
  <Direction id="id3" Name="Z direction" Direction_ratios="0 0 1"/>
  <Direction id="id4" Name="X direction" Direction_ratios="1 0 0"/>
</iso_10303_28_terse>
```

### 1.4.1 Sample Part 28 Program

The program below reads a STEP file (Part 21 or Part 28) and writes it back out to a different name and format. Including **rose_p28.h** and calling **rose_p28_init()** forces the program to recognize Part 28 as well as Part 21.

```
#include <rose.h>
#include <rose_p28.h>  // bring in Part 28 support
```

```
void main(int argc, char ** argv)
{
    rose_p28_init();   // call at start of program
    if (argc != 4) {
        printf ("usage: format <p21|p28|p28-raw> <in-name> <out-name>\n");
        return;
    }

    const char * fmt = argv[1];
    const char * in_name = argv[2];
    const char * out_name = argv[3];

    RoseDesign * d = ROSE.findDesign(in_name);
    if (d) {
        d->path (out_name);
        d->format(fmt);
        d->save();
    }
    else printf ("ERROR: Could not read design '%s'\n", in_name);
}
```

To build this sample program, compile normally, but add the **p28e2** library to the link line as shown in the following commands:

```
On UNIX and MacOS
% CC -I$ROSE_INCLUDE format.cxx -L$ROSE_LIB -lrose -lp28e2

On Windows
> cl /I"%ROSE_INCLUDE%" format.cxx /LIBPATH:"%ROSE_LIB%" rose.lib p28e2.lib
```

## 1.4.2 Part 28 XML Browser

ST-Developer makes it simple to examine STEP XML files in any web browser with the Part 28 to HTML conversion tool. On Windows, this tool integrates into the Explorer so you can open a file in a web browser by simply right clicking on it and selecting “Browse P28 XML” as shown below.

On other platforms just run the **p28html** command-line tool to prepare HTML for your browser as shown below:

```
% p28html datafile.p28    ==> writes datafile.html in the current dir
% firefox datafile.html
```

All of the entity instances are linked, so you can move through the file simply by clicking on references.

As shown in the figure above, clicking on any instance in the file pops up a window

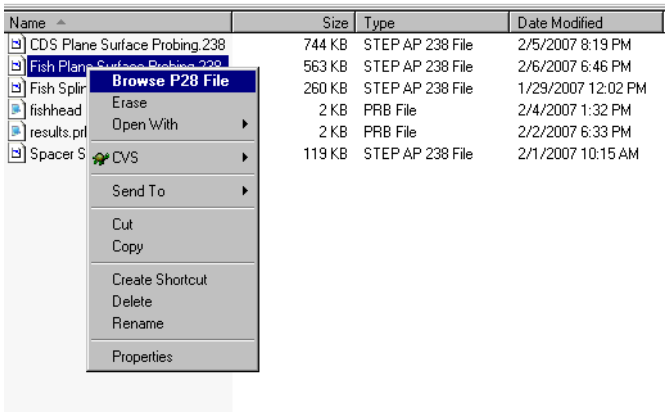



**Figure 1.2 —** Browse P28 XML

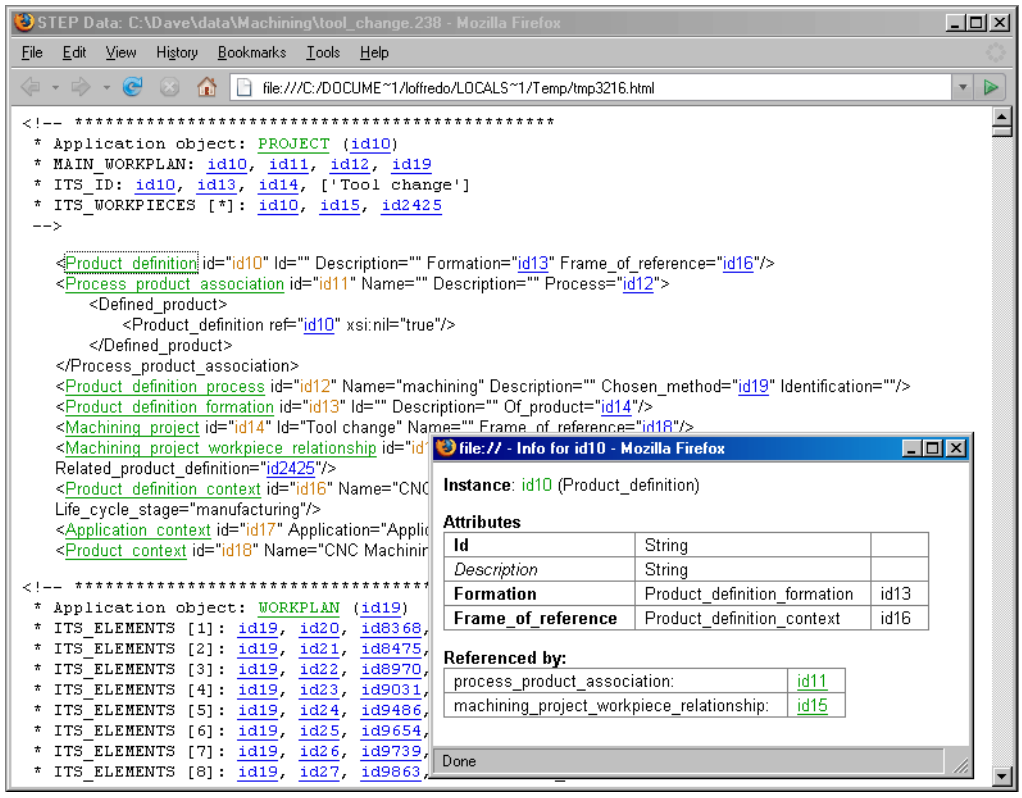



**Figure 1.3 —** STEP P28 XML In a Web Browser

showing all attributes, their types and their values, even unset attributes. **Required attributes** are shown in bold, while *optional attributes* are italicized. You will also see a complete USEDIN() listing of objects that refer to the instance, and can quickly jump to any of them.

Some simple validation checks are done and objects that have missing required attributes are hilighted in red on the main page, along with each offending attribute in

the attributes popup window.

To make it easier for others to examine the XML data in the new Part 28 format, the **p28view** and **p28html** executables are not locked, so that you can distribute them for non-commercial use to any third party who might be interested in the new data.

## 1.5 ST-Developer Message Window

ST-Developer for Windows now contains the **rose_logwin** extension to simplify the display of status and debugging information from GUI applications. The extension creates a window to hold a cumulative log of status messages issued by the ROSE library or your application. The messages can be cleared, copied, and saved, and the window can be dismissed when not needed.

The the **rose_logwin** extension provides its own Windows message pump in a dedicated thread, and can be used by any type of ROSE C++ application: MFC, raw Windows API, or even a console application. The screendump below shows the message window and some diagnostic messages from a sample application.

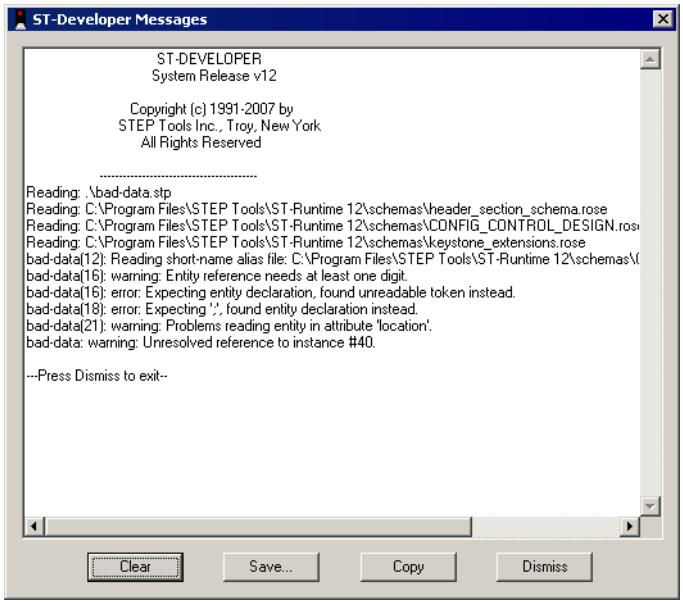



**Figure 1.4 —** ST-Developer Message Window

To enable this functionality, simply link against the **rose_logwin.lib** library, include the **rose_logwin.h** file, then call **rose_logwin_init()** at the start of your program to register the message window with the ROSE library.

After this call, all ROSE messages are redirected to the window. No other changes are required for your application. Calling **rose_logwin_wait()** at the end of your program will block until the user dismisses the message box. This gives the user a chance to view the messages, particularly in short-lived programs.

The following example shows how to initialize the message window:

```
#include <rose.h>
#include <rose_logwin.h>

int main (int argc, char ** argv)
{
    rose_logwin_init();
    /* your code */
    rose_logwin_wait();
}
```

The **roselog** DLL must be in your path for the log window to appear. It is installed in the ST-Developer **bin** directory, which should be in your path. If the DLL cannot be found, **rose_logwin_init()** will return zero and the default error handling used.

If you want to display your own status messages in the window, simply call one of the ROSE error reporting functions:

```
ROSE.message ("a status message");  // supressed if ROSE.quiet(1);
ROSE.warning ("a warning message");
ROSE.error ("an error message");    // will exit by default
```

## 1.6 ROSE Library Enhancements

We have added a number of improvements to ROSE to simplify STEP data processing and improve the performance of applications. Consult the *ROSE Library Reference Manual* for a complete description of these capabilities.

### 1.6.1 Binary Type Support

Until recently, the EXPRESS “binary” primitive type has never been used by any of the STEP application protocols, but the new AP221 and IFC 2x3 schemas define entities that contain a binary attribute or list of binary values. Previous versions of ST-Developer supported individual values of the binary primitive type but not aggregates of binary.

We have changed the in-memory representation of the EXPRESS "binary" type so that these values are now stored as strings. This simplified use of binary values and allowed us to implement aggregates of binary. The EXPRESS compiler has been updated to generate the new C++ representation, and any existing classes that used binary should be regenerated.

The C++ type used for getting and putting a binary value is **RoseBinarySTR** and values are stored in the string encoding used by the Part 21 file format. We provide the **RoseBinaryObject** helper class to handle bit and byte level access to the data.

The example below shows an object with a binary attribute called "binary_att" and puts, gets, and manupulates some binary data. Consult the *ROSE Library Reference Manual* for a complete description of the **RoseBinaryObject** class.

```
SomeEntity * obj;


// Set attribute to hex 0xFFFF
obj-> binary_att ("0FFFF");

// print the value back out again
printf ("Binary Value %s\n", obj-> binary_att());

// use helper class for bit and byte level access
RoseBinaryObject helper (obj-> binary_att());

helper.putBit (8, 0)        // changes to 0xFFEF
helper.putByte (1, 0xA7)    // changes to 0xA7EF
helper.putWord (1, 0x1234)  // resizes to 0x1234A7EF

// Update the attribute to with the modified value
obj-> binary_att (helper.asString());
```

## 1.6.2 Other Library Changes

Changed the **STR** typedef to **RoseSTR** to avoid potential conflicts between packages. The old definitions can still be used, but can also be turned off by defining the **ROSE_DISABLE_COMPAT_DEFS** symbol on the command line.

Added the following convenience functions to **RoseAttribute** to simplify type testing. There were already tests for string, double, boolean, and other primitive types, but the following functions complete the range of possible types.

```
RoseAttribute::isAggregate()
RoseAttribute::isEntity()
RoseAttribute::isEnumeration()
RoseAttribute::isSelect()
```

```
RoseAttribute::isSimple()
```

Added the **rose_is_system_schema()** function so that applications can distinguish between system schemas that are used internally by the ROSE library and schemas that represent an application protocol or other public schema. This is useful when deciding which schemas to write in the header of a STEP file.

```
int rose_is_system_schema(RoseDesign *);
```

Added the **RoseEidCursor** class to simplify traversing a design in numeric order by the Part 21 file entity identifiers. The following code fragment shows how this can be used:

```
RoseDesign * d;
RoseEidCursor objs (d);
RoseStructure * obj;

while (obj = objs.next()) {
    // do something
}
```

### 1.6.3 Deprecated Functions

Many of the functions in the **RoseInterface** object have been deprecated. These functions are simply aliases of **RoseDesign** functions and will be removed in a future release.

```
RoseInterface::addName()
RoseInterface::addSchema()
RoseInterface::designName()
RoseInterface::display()
RoseInterface::findDomain()
RoseInterface::findObject()
RoseInterface::findObjects()
RoseInterface::format()
RoseInterface::nameTable()
RoseInterface::pnewInstance()
RoseInterface::removeDesign()
RoseInterface::removeName()
RoseInterface::rootObject()
RoseInterface::saveDesign()
RoseInterface::saveDesignAs()
RoseInterface::setDesignName()
RoseInterface::setFormat()
RoseInterface::setRootObject()
RoseInterface::useSchema()
```

In addition, the following type aliases have been deprecated and will be removed in

a future release. The previously deprecated **BaseManager** alias for **RoseManager** has been removed.

```
STR         ==> use RoseSTR
ArrayOfSTR  ==> use ArrayOfString
BagOfSTR    ==> use BagOfString
ListOfSTR   ==> use ListOfString
SetOfSTR    ==> use SetOfString
DictionaryOfSTR  ==> use DictionaryOfString
```

# 2 ST-Developer Quick Tour

## 2.1 Getting Started

Once you have installed ST-Developer as described in the following chapters, you can start programming immediately, just by linking against the appropriate library. ST-Developer ships with pre-built C++ and Java class libraries for all common information models. Just start with one of the sample programs we provide for each library, then modify as needed.

Bring up the ST-Developer online documentation by selecting the **STEP Tools | Documentation** entry on the Windows Start menu or by directing your web browser to the following file in the ST-Developer installation directory:

```
file:/<stdev_install_dir>/docs/index.html
```

The online documentation has hyperlinked versions of all manuals, PDF versions for printing, and pages for each information model. The model pages contain one or more sample programs with sample output, web-browsable and plain text versions of the EXPRESS schema, lists of classes, and any recommended practices documents. The AP214 page is shown in Figure 2.1.

For each sample program you will see a Windows project link and a MacOS/Unix project link. Right-click on the link to save the project zip file. Unpack the zip file and then either open the project file within with Visual Studio (Windows) or compile the application by running **make** (MacOS/Unix).

The sample programs contain utilities for creating STEP units, measure values, contexts, and other common data that you can reuse for your own applications.

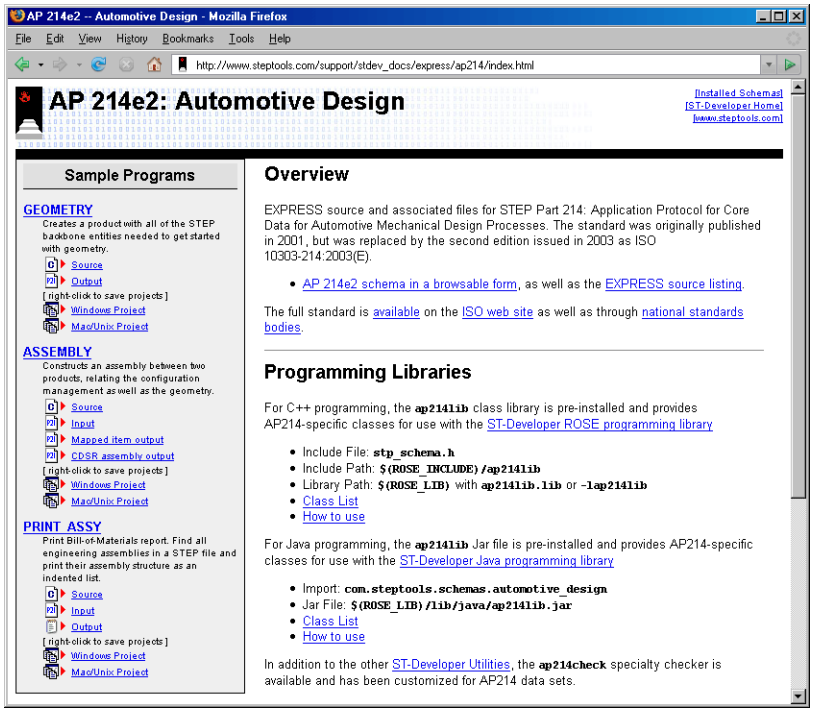



**Figure 2.1 —** Sample Programs and Schema Documentation

### 2.1.1 Windows Control Panel

On Windows, you can also start the ST-Developer Control Panel by selecting the **STEP Tools | STEP Tools Control Panel** entry on the Start menu. This is a graphical front-end to the ST-Developer command line tools as shown in Figure 2.2.

When a tool is run from the control panel, any text printed by the tool will appear in the **Output Pane**. You can browse this output and copy it onto the clipboard for later use.

When a tool such as the EXPRESS compiler produces an error message with a file name and line number, you can double click on this message in the output box. The control panel will start the Windows Notepad editor, load the file into it and highlight the line where the error or warning condition was found.

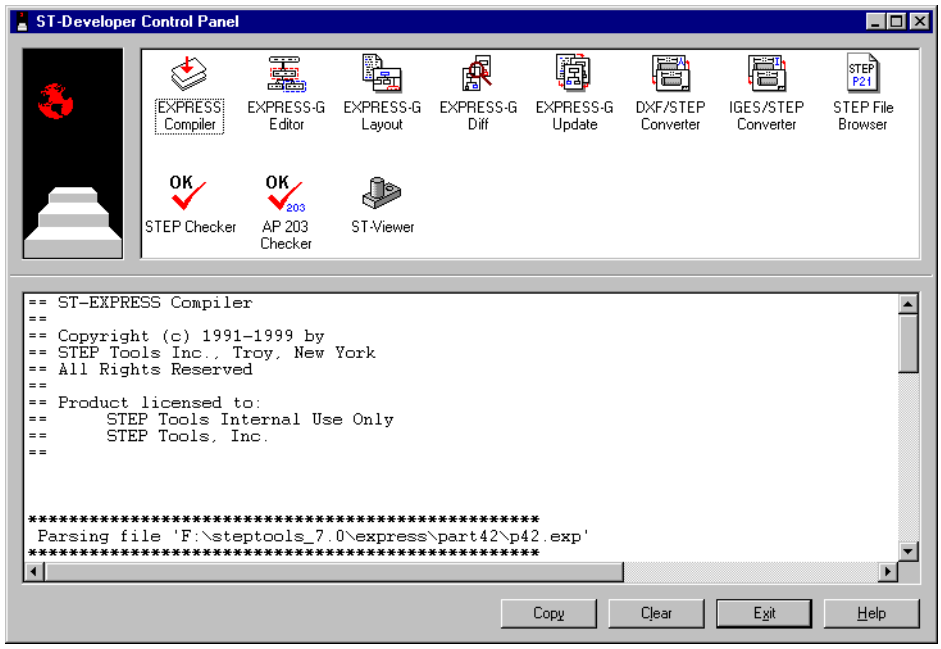



**Figure 2.2 —** ST-Developer Control Panel

## 2.2 What is in ST-Developer?

ST-Developer is a set of software tools to build, operate and maintain your STEP, IFC, CIS/2 and EXPRESS-defined tools, translators and databases. It contains programming bindings for C++, C, and Java, plus tools for testing data sets against verification rules and constraints, browsing through the contents of your data sets, building information models, and more.

### 2.2.1 Programming Environments

ST-Developer ships with pre-built STEP AP Class Libraries, ROSE C++ and Java class libraries for many STEP APs and other models like CIS/2 and IFC. You can start programming immediately, just by linking against the appropriate library. The ST-Developer programming libraries are:

- **ROSE C++ Library** — For demanding CAD and data exchange applications. C++ classes generated from EXPRESS schemas mean fast access, and strong

compiler type checking helps to build reliable applications. The ROSE library provides many advanced object search and traversal features such as USEDIN, early and late-bound access, greater control over STEP physical file handling, and AP interoperability extensions. Refer to the *ROSE Library Reference Manual* for more information.

In addition, ST-Developer ships with several extension libraries for ROSE applications, such as the Part 28 library for reading and writing XML, the Part 21 filter library for working on very large STEP Part 21 files out of memory, and the ROSE Log Window library for status messages in GUI applications.

- **SDAI C Library** — Build applications using a small set of straightforward C functions to manipulate STEP data. No application protocol-specific structures or classes are used. Refer to the *SDAI C Library Reference Manual* for more information on SDAI C applications.

- **ST-Developer for Java Library** — Using the EXPRESS compiler, you can build applications using Java classes. Strong compiler type checking helps to build reliable applications. Refer to the *ST-Developer Java Library Reference Manual* for more information.

- **STEP AP and other Schema Libraries** — ST-Developer ships with pre-built ROSE C++ and Java class libraries for many STEP APs and other information models like CIS/2 and IFC. You can start programming immediately, just by linking against the appropriate library as described above in **Getting Started** (Section 2.1, pp. 15).

### 2.2.2 Conformance Testing, Browsing, and Editing Tools

View and browse STEP data sets then check for adherence to the STEP Application Protocols using these tools. The checking tool evaluate a data set against the structures, EXPRESS constraints, and usage recommendations to verify it is correct.

- **ST-Viewer** — 3D view of STEP geometry and assemblies. (Windows only)

- **ap203check** — Check AP-203 data sets using a custom tool.

- **ap209check** — Check AP-209 data sets using a custom tool.

- **ap214check** — Check AP-214 data sets using a custom tool.

- **apconform** — Check any data set against the constraints in an EXPRESS information model using an EXPRESS interpreter.

- **p28view/p28html** — Examine STEP Part 28 XML files in any web browser.
- **stepedit** — Examine, edit, and check STEP Part 21 files. (UNIX only)
- **stepbrws** — Examine STEP Part 21 files. (Windows only)
- **stepclean** — Remove redundant information from STEP files.

### 2.2.3 Information Modeling Tools

These EXPRESS tools process the STEP schema definitions and produce useful things from them. The EXPRESS compiler checks schemas quite thoroughly and can generate C++ or Java classes, SDAI data-dictionaries, and HTML. The EXPRESS-G tools construct EXPRESS-G diagrams from any EXPRESS information model and display, rearrange, and print them using a graphical editor.

- **expfront** — Unified EXPRESS compiler that can check schemas, generate classes for C++ and Java, merge short-form schemas into a single long-form, and perform other useful functions.
- **express2cxx** — Generate C++ classes from EXPRESS.
- **express2java** — Generate Java classes from EXPRESS.
- **express2html** — Convert EXPRESS Schemas to HTML.
- **expgedit** — EXPRESS-G diagram editor.
- **express2expg** — EXPRESS compiler to generate EXPRESS-G diagrams.
- **expgupdate** — Update EXPRESS-G diagram.
- **expgdiff** — EXPRESS-G difference utility.
- **expg2ps** — EXPRESS-G to PostScript converter.
- **expg2hpgl** — EXPRESS-G to HP-GL converter.
- **expginfo** — Get information from an EXPRESS-G diagram.
- **expgsetopts** — Set options on an EXPRESS-G diagram.

### 2.2.4 File Conversion Tools

The file format converters transform IGES or DXF files into STEP exchange files and vice-versa. This allows ST-Developer applications to access IGES and DXF data in the same way as data described by other EXPRESS information models. They do not perform geometric transformation from IGES/DXF to STEP Part 42 structures, but they can serve as the front end for applications that do.

- **dxf2step** — Convert an AutoCAD DXF file to STEP Part 21.
- **iges2step** — Convert an IGES file to STEP Part 21.
- **step2dxf** — Convert a STEP file to DXF.
- **step2iges** — Convert a STEP file to IGES.

### 2.2.5 Project Tools

Windows versions of ST-Developer contain the ST-Developer Control Panel and Microsoft Visual Studio wizards and plug-ins to speed application development. All platforms contain the extclass. We have provided several additional tools to make the management of these classes somewhat easier.

- **extclass** — Add hooks for extensions to a generated C++ class.
- **extall** — Add hooks to many C++ classes as needed.
- **mkmakefile** — Create makefile for generated C++ classes.
- **rose** — Utilities for ROSE Working Form and Part 21 files.

# 3 Migrating from Earlier Versions

## 3.1 From ST-Developer v11

You should not require any additional source code changes to software previously working with ST-Developer v11. Since the binary layout of the libraries may have changed, you must completely recompile (i.e. remove all object files) any applications you have developed.

If your application used the EXPRESS "binary" type, you may need to update some source code, since the ROSE API has changed, however, this is unlikely since none of the STEP Application Protocols have ever used this feature.

To begin using new features, such as the Part28 XML reader and writer, or the ROSE logging window, follow the instructions given in the previous chapter.

## 3.2 From ST-Developer v10

Since additional methods have been added to the rose library, you must completely recompile (i.e. remove all object files) any applications you have developed.

If you are using either the **RoseP21Parser::add_schema_fn** or the **RoseP21Writer::schema_name_fn** hook functions, you will need to change the declarations of your functions to accommodate the new arguments. Your code will not compile if you use the old prototypes, so you will know immediately if this affects you.

If you are including **RoseP21Lex.h**, **RoseP21Parser.h** or **RoseP21Writer.h**, you no longer need to do so. These are now brought in automatically by **rose.h**.

## 3.3 From ST-Developer v9

In addition to the changes described above, the **stepi18n** library has been merged into the **rose** library, so any references to it should be removed from your makefiles.

All runtime support files required by ST-Developer applications are now in the re-distributable ST-Runtime package. The contents of the **system_db** directory have been either moved to ST-Runtime or to the **lib** subdirectory (for items only used by ST-Developer).

On Windows, ST-Runtime is provided as a stand-alone Windows Installer (msi) and as an installer merge module. You can merge ST-Runtime into your installer or just ship the ST-Runtime installer along with your application, and let the user install the two independently. On UNIX and MacOS, ST-Runtime is provided as zip file.

For more information, see **Packaging Applications for Distribution** in the Rose Library Reference Manual.

The sortnames files (***.nam**) now use the same syntax as the AP documents (long-name first, then shortname). If you have any of your own *.nam files, you should update them to the new syntax. Shortname files for all of the STEP APs are already present in ST-Runtime.

## 3.4 From ST-Developer v8 and earlier

You should not require any additional API changes with ST-Developer v8 or earlier. If you are using any compiled schemas files that are not already present in the ST-Runtime package, you must regenerate them with the EXPRESS compiler.

In ST-Developer v7 and earlier, the C++ classes for aggregates were generated in separate files, but now they are kept with their base type. You may need to update your makefiles or VC projects for the new set of source files.

# 4 Windows Installation

## 4.1 Platform Requirements

ST-Developer supports Intel x86 machines with Windows XP/Vista and software development with Visual Studio 2005 (VC 8.0) as well as the older Microsoft Visual C++ v6.0 compiler. Consult the **README.TXT** file on the distribution media for exact disk space requirements and other details.

You must install ST-Developer using an account with administrator privileges. After the installation is complete, you can run the tools under any user account. Under Windows 2000, you must have "Power User" privileges.

## 4.2 Installation Procedure

### 4.2.1 Remove Earlier Versions

1. If you are upgrading from a v10 or earlier installation with Visual Studio plug-ins, use the **Start | Programs | ST-Developer | Visual C++ Integration** tool to "unintegrate" the Visual Studio plug-ins. Do this for each ST-Developer user.

2. Go to the **Start | Settings | Control Panel** menu item. Select **Add/Remove Programs**, then choose the ST-Developer entry from the list of packages.

### 4.2.2 Install the New Version

1. Insert the ST-Developer CD-ROM into your drive. If the installer does not start automatically, double click on **stdev.msi.**

   Some pre-XP machines may not recognize the **.msi** extension, or may complain that the package cannot be installed by the Windows Installer service. If either of these happen, just install the Windows MSI 2.0 upgrade below.

   Windows 2000/NT: Double click on **InstMsiW.exe**.
   Windows 98 or ME: Double click on **InstMsiA.exe**.

2. The installer will guide you through the installation process. If you wish to omit parts of ST-Developer, or specify where it is installed, select a custom installation when prompted, otherwise select typical installation.

3. Request a license key when prompted. Click **Register** and follow the instructions in the web form. If you wish to request a key later, select **STEP Tools | Request License Key** on the Start menu. We will send the key by e-mail and you can install it as described in **Software License Keys** (Section 7, pp. 37).

ST-Developer is now installed. You can verify the environment settings on your machine and test the ST-Developer tools as described in the following sections.

### 4.2.3 Visual Studio and Environment Settings

The ST-Developer installer sets several environment variables used by the Visual Studio project files and makefiles for the sample programs to find programming libraries, header files, and tools.

To check these, start a command prompt and type **set**, or bring up the **Environment** tab of the **System** control panel. Verify that each has an accurate value:

**%ROSE%** . . . . . ST-Developer installation directory, also set under **Software\STEP Tools, Inc.\ST-Developer** in the registry.

**%ROSE_BIN%** . . . . . Directory containing ST-Developer executables. Usually **%ROSE%\bin**. This is added to your **%PATH%** variable.

**%ROSE_INCLUDE%** . Directory containing the ST-Developer C++ header files. Usually **%ROSE%\include**. Used by project files.

**%ROSE_LIB**% . . . . . Directory containing the ST-Developer C++ libraries. Usually a subdirectory under **%ROSE%\lib**. Used by project files.

**%ROSE_CONFIG%** . . Location of the makefile configuration file. This is a file in the **%ROSE%\config** directory that defines settings for the demo makefiles.

ST-Developer includes libraries built for newer and older versions of Visual C++ as well as multi-thread, static (**/MT** flag) and multi-thread, dynamic (**/MD** flag) link conventions. By default, the **ROSE_LIB** variable points to the libraries for Visual Studio 2005 with the **/MD** flag. To usea different version, change **ROSE_LIB** as below:

```
ST-Developer for 32bit Windows:
<stdev-install-dir>\lib\i86_win32_vc8_md     Visual Studio 2005 with /MD
<stdev-install-dir>\lib\i86_win32_vc8_mt     Visual Studio 2005 with /MT
<stdev-install-dir>\lib\i86_win32_vc6_md     Visual C++ 6 with /MD
<stdev-install-dir>\lib\i86_win32_vc6_mt     Visual C++ 6 with /MT

ST-Developer for 64bit Windows:
<stdev-install-dir>\lib\x64_win64_vc8_md     Visual Studio 2005 with /MD
<stdev-install-dir>\lib\x64_win64_vc8_mt     Visual Studio 2005 with /MT
```

You can also specify one of these explicitly as a **/LIBPATH** in your project settings if you need alternate versions.

If you write code that executes in multiple threads, be aware that the ST-Developer libraries are **not** reentrant. You must ensure that only one thread calls an ST-Developer library function at any point in time.

### 4.2.4 Verify the Installation

You can test the various components of ST-Developer using the following steps:

1. Select **STEP Tools | Documentation** on the Start menu to open the ST-Developer online manuals. The manual home page is located at the following URL within the installation directory:

   ```
   file:/<stdev_install_dir>/docs/index.html
   ```

2. Open a command prompt and use the **rose** file utility to list the installed system resource files. This verifies that the path, registry, and environment settings are properly configured. If you have problems, make sure that the **%ROSE_BIN%** directory is in your search path.

```
C:\> rose ls
```

It should return something like the following:

```
Total number of designs::  100

 In C:\Program Files\STEP Tools\ST-Runtime 12\schemas\*.*
 ?      ap239_product_life_cycle_support_mim_lf.rose
        ap239_product_life_cycle_support_mim_lf_EXPX.rose
 ?      associative_draughting.rose
        associative_draughting_EXPX.rose
 ?      automotive_design.rose
        automotive_design_EXPX.rose
 ?      building_design_schema.rose
        building_design_schema_EXPX.rose
        [ . . . files ommitted . . . ]
```

**3.** Go to the Start menu and open the ST-Developer online manuals. You will see a list of links to pre-installed schemas (AP201, AP202, etc). Go to one of them and compile one of the sample programs.

For each sample program you will see a Windows project link. Right-click on the link to save the project zip file. Unpack the zip file and open the project file within with Visual Studio.

If you prefer to work from the command line, there is also a makefile within the zip that you can build with **nmake**. If you may need to first run the **vcvars32.bat** file supplied with Visual Studio to put the **cl** compiler into your command line search path.

# 5 MacOS Installation

## 5.1 Platform Requirements

ST-Developer supports the following Macintosh platforms. You should also consult the **README** file on the CD for exact disk space requirements and other installation details

- ST-Developer for MacOS PowerPC requires MacOS X version 10.4 (Tiger) and contains libraries built for the XCode v2.0 tools.
- ST-Developer for MacOS Intel 32bit requires MacOS X version 10.4 (Tiger) and contains libraries built for the XCode v2.3 tools.
- ST-Developer for MacOS Intel 64bit requires MacOS X version 10.5 (Leopard) and contains libraries built for the XCode v2.3 tools.

The EXPRESS-G Editor and STEP browser are X11 GUI applications that require the X Window System. The XCode and X11 packages are available for free download on Apple's web site at **www.apple.com** and may also be available on the operating system install disks.

## 5.2 Installation Procedure

1. Insert the ST-Developer CD-ROM. Open a terminal window (Applications/ Utilities/Terminal) and **cd** to your desired installation directory.

   Use **unzip** to extract files **/Volumes/stdev-12-macosx/STDEV.ZIP**. This will create a directory called **steptools_12**.

   ```
   % cd /usr/local
   % unzip /Volumes/stdev-12-macosx/STDEV.ZIP
   ```

2. Go into the **steptools_12** directory and run the **stdev_install** script. This will verify the install location as well as some information about the C++ compiler.

   ```
   % cd steptools_12
   % ./stdev_install
   ```

3. Request, and then install a license key as described in **Software License Keys** (Section 7, pp. 37). We will send the key by e-mail.

ST-Developer is now installed. You can verify the environment settings on your machine and test the ST-Developer tools as described in the following sections.

### 5.2.1 Environment Settings

The install script creates a **Rose_Logicals** file in the ST-Developer installation directory with settings specific to your site. Check the file to make sure the following environment variables are set correctly:

**ROSE** . . . . . ST-Developer installation directory.

**ROSE_BIN** . . . . . Directory containing executables. Usually **$ROSE/bin**. Put this in your shell search path.

**ROSE_INCLUDE** . . . . Directory containing the ST-Developer C++ header files. Usually **$ROSE/include**. Used by makefiles.

**ROSE_LIB** . . . . . Directory containing the ST-Developer C++ libraries. Usually a subdirectory under **$ROSE/lib**. Used by makefiles.

**ROSE_CONFIG** . . . . . Makefile configuration file in **$ROSE/config** that defines settings for your compiler and build options. This file is included by the makefiles for all sample programs.

**ROSE_RUNTIME** . . . . Optional. Location of the ST-Runtime support files. If not set, tools look for **$ROSE/runtime**.

**ROSE_LICENSE**. . . . . Optional. An alternate location for the license key file. If not set, tools look for **$ROSE/license**.

ST-Developer may include versions of the libraries built for different compilers or link options. The **$ROSE_CONFIG** and **$ROSE_LIB%** variables are set by the install script to your default choice, but you can change them later if you need to use alternate library versions.

### 5.2.2 Verify the Installation

You can test the various components of ST-Developer using the following steps:

1. Open the ST-Developer online manuals with your web browser. The manual home page is located at the following URL within the installation directory:

   ```
   file:/<stdev_install_dir>/docs/index.html
   ```

2. Start a new shell and evaluate the **Rose_Logicals.sh** file to set the environment variables described above. Add **$ROSE_BIN** to your search path. Make sure these are always set before you use any of the ST-Developer tools.

   ```
   % . Rose_Logicals.sh
   % export PATH=$ROSE_BIN:$PATH
   ```

3. Use the **rose** file utility to list the installed system resource files. This verifies that the path and environment settings are properly configured. If you have problems, make sure that the **$ROSE_BIN** directory is in your search path

   ```
   % rose ls
   ```

   It should return something like the following:

   ```
   Total number of designs::  100

    In C:\Program Files\STEP Tools\ST-Runtime 12\schemas\*.*
    ?      ap239_product_life_cycle_support_mim_lf.rose
           ap239_product_life_cycle_support_mim_lf_EXPX.rose
    ?      associative_draughting.rose
   ```

```
        associative_draughting_EXPX.rose
?       automotive_design.rose
        automotive_design_EXPX.rose
?       building_design_schema.rose
        building_design_schema_EXPX.rose
        [ . . . files ommitted . . . ]
```

**4.** Open the ST-Developer online manuals as described above. You will see a list of links to pre-installed schemas (AP201, AP202, etc). Go to one of them and compile one of the sample programs.

For each sample program you will see a Unix project link. Right-click on the link to save the project zip file. Unpack the zip file and compile the application by running **make**.

# 6 UNIX Installation

## 6.1 Platform Requirements

This section describes operating system and compiler details for supported UNIX platforms. Consult the **README** file on the CD for exact disk space requirements and other installation details.

Some C++ compilers require flags or other special treatment. This is handled by makefile definitions in the **$ROSE/config** directory. Makefiles include one of these configuration files through the **$ROSE_CONFIG** environment variable. The settings are normally handled by the ST-Developer installation script.

### 6.1.1 Linux Notes

ST-Developer for Linux will run on a wide variety of Linux distributions. The basic requirement is **glibc** v2.3.2 or later and a compatible version of GCC. This should be satisfied by any distribution compliant to Linux Standard Base (LSB) either 2.x or 3.x. The executables will work on older distros running glibc 2.3.2 (such as RedHat 9 and RHEL3, Fedora Core, SuSE 8.2/9.0, Debian 3.1, Gentoo 1.4/2004.1, Linspire 4.5/5.0), newer versions running glibc 2.3.4 (RHEL 4, Fedora 4, SuSE 9.3, Gentoo 2004.3, Ubuntu 5.10), and glibc 2.5 (RHEL 5, Fedora 6-7, SuSE 10.2, Gentoo 2007, Ubuntu 7.04). To see what your system has, run "**ldd --version**".

The ST-Developer libraries are built for use with several versions of GCC. If you look under the ST-Developer **lib** directory you will see several versions. The linking conventions change between GCC 3.2/3.3 and the latest GCC 3.4/4.x compilers, so

make sure that you link against the correct ones for your compiler. To see what GCC version your system has, run "**gcc --version**".

### 6.1.2 Hewlett Packard PA-RISC / HPUX Notes

ST-Developer for Hewlett Packard PA-RISC requires HP-UX 11. The installation is compatible with the 9000 Series 700 PA-RISC 1.1 machines as well as the newer C-series PA-RISC 2.0 machines. The C++ programing libraries are built for use with the HP ANSI C++ compiler (**aCC**).

### 6.1.3 Silicon Graphics / IRIX Notes

ST-Developer for Silicon Graphics can be used with IRIX 6.2 and later. The C++ programing libraries are built for use with the SGI MIPSpro C++ compiler.

### 6.1.4 Sun SPARC Notes

ST-Developer for Sun SPARC can be used with Solaris v7 and later. The distribution contains versions of the C++ programing libraries built for use with Forte and Sun Studio C++ (Sun Workshop 5.0, Forte Developer 6, Sun Studio 7 to 11), and the GNU GCC v3.4/4.x compilers.

## 6.2 Installation Procedure

1. Mount the ST-Developer CD-ROM and **cd** to your desired installation directory. See to **Mounting CD-ROMs on Various Unix Systems** (Section 6.3, pp. 35) if you need assistance mounting the disk.

   Use **unzip** to extract from **/cdrom/stdev.zip**. This will create a directory called **steptools_12**. A copy of the **unzip** program is included on the CD in the unzip directory. If several zip files are present, consult the **README** file on the CD to see which one to unpack.

   In the following example, replace **/usr/local** with the install location and replace **/cdrom** with the actual CD mount point.

```
% cd /usr/local
% unzip /cdrom/stdev.zip
```

2. Go into the **steptools_12** directory and run the **stdev_install** script. This will verify the install location as well as some information about the C++ compiler.

```
% cd steptools_11
% ./stdev_install
```

3. Request, and then install a license key as described in **Software License Keys** (Section 7, pp. 37). We will send the key by e-mail.

ST-Developer is now installed. You can verify the environment settings on your machine and test the ST-Developer tools as described in the following sections.

### 6.2.1 Environment Settings

The install script creates a **Rose_Logicals** file in the ST-Developer installation directory with settings specific to your site. Check the file to make sure the following environment variables are set correctly:

**ROSE** . . . . . ST-Developer installation directory.

**ROSE_BIN** . . . . . Directory containing executables. Usually **$ROSE/bin**. Put this in your shell search path.

**ROSE_INCLUDE** . . . . Directory containing the ST-Developer C++ header files. Usually **$ROSE/include**. Used by makefiles.

**ROSE_LIB** . . . . . Directory containing the ST-Developer C++ libraries. Usually a subdirectory under **$ROSE/lib**. Used by makefiles.

**ROSE_CONFIG** . . . . . Makefile configuration file in **$ROSE/config** that defines settings for your compiler and build options. This file is included by the makefiles for all sample programs.

**ROSE_RUNTIME** . . . . Optional. Location of the ST-Runtime support files. If not set, tools look for **$ROSE/runtime**.

**ROSE_LICENSE**. . . . . Optional. An alternate location for the license key file. If not set, tools look for **$ROSE/license**.

ST-Developer may include versions of the libraries built for different compilers or

link options. The **$ROSE_CONFIG** and **$ROSE_LIB** variables are set by the install script to your default choice, but you can change them later if you need to use alternate library versions.

### 6.2.2 Verify the Installation

You can test the various components of ST-Developer using the following steps:

1. Open the ST-Developer online manuals with your web browser. The manual home page is located at the following URL within the installation directory:

   ```
   file:/<stdev_install_dir>/docs/index.html
   ```

2. Start a new shell and evaluate the **Rose_Logicals** (csh) or **Rose_Logicals.sh** (bash) file to set the environment variables described above. Add **$ROSE_BIN** to your search path. Make sure these are always set before you use any of the ST-Developer tools.

   ```
   % source Rose_Logicals
   % set path=( $ROSE_BIN $path )

   % . Rose_Logicals.sh
   % export PATH=$ROSE_BIN:$PATH
   ```

3. Use the **rose** file utility to list the installed system resource files. This verifies that the path and environment settings are properly configured. If you have problems, make sure that the **$ROSE_BIN** directory is in your search path

   ```
   % rose ls
   ```

   It should return something like the following:

   ```
   Total number of designs::  100

    In C:\Program Files\STEP Tools\ST-Runtime 12\schemas\*.*
    ?      ap239_product_life_cycle_support_mim_lf.rose
           ap239_product_life_cycle_support_mim_lf_EXPX.rose
    ?      associative_draughting.rose
           associative_draughting_EXPX.rose
    ?      automotive_design.rose
           automotive_design_EXPX.rose
    ?      building_design_schema.rose
           building_design_schema_EXPX.rose
           [ . . . files ommitted . . . ]
   ```

4. Open the ST-Developer online manuals as described above. You will see a list of links to pre-installed schemas (AP201, AP202, etc). Go to one of them and compile one of the sample programs.

   For each sample program you will see a Unix project link. Right-click on the link to save the project zip file. Unpack the zip file and compile the application by running **make**.

## 6.3 Mounting CD-ROMs on Various Unix Systems

STEP Tools CDs use an ISO 9660 filesystem, possibly with Rock Ridge extensions. The procedure for mounting CD-ROMs varies, so we have included instructions for some common platforms:

### 6.3.1 Linux

Simply insert the disk. On most modern Linux distros, it will be automatically mounted under **/mnt/cdrom.** If you have a different setup, you may need to manually mount the disk using a command like the following.

```
% mount -r -t iso9660 /dev/hdc /cdrom   # IDE CD
% mount -r -t iso9660 /dev/scd0 /cdrom  # SCSI CD
% umount /cdrom                        # to unmount the disk
```

### 6.3.2 Hewlett Packard PA-RISC / HP-UX

Using the **sam** system administration tool, invoke **sam** as root, navigate to **Disk and File Systems**, and then to **File Systems**. Under **Actions | Add Local File System | Not Using The Logical Volume Manager**, you will find a dialog box that mounts a CD as a one-time event.

If you know the device name, you can also mount the disk manually using:

```
% mount -r -F cdfs /dev/dsk/c1t2d0 /cdrom
% umount /cdrom
```

Where **c1t2d0** is the actual device name of the CD-ROM drive.

### 6.3.3 Silicon Graphics/IRIX

Simply insert the disk. The system will automatically mount the disk as **/CDROM**. To unmount and eject the disk:

```
% eject /CDROM          # to unmount the disk
```

### 6.3.4 Sun SPARC/Solaris

Simply insert the disk. The volume manager will automatically mount it under the **/cdrom** directory as:

```
/cdrom/<volumename>     (such as /cdrom/cdrom0)
```

To unmount and eject the disk:

```
% eject cd
```

# 7 Software License Keys

## 7.1 Requesting a License Key

The ST-Developer tools, such as the EXPRESS compiler or AP203 Checker, require a license key. Applications that you build using the ST-Developer libraries do not need a key and can be distributed without restriction.

To request a key, fill in the request form at the URL below. Selecting **STEP Tools | Request License Key** on the Windows Start menu will take you to this form and fill in most information. The Windows installer will also take you to this form.

```
http://www.steptools.com/register/
```

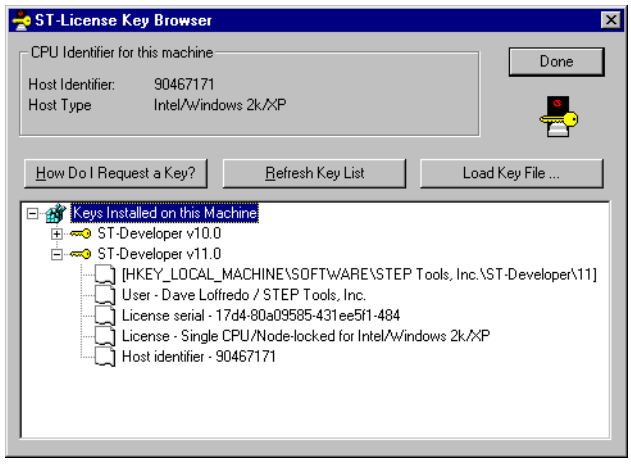



**Figure 7.1 —** ST-License Key Browser

On Windows, the ST-License Key Browser shown in Figure 7.1 will display the host identifier of your machine (an 8-digit hex number) as well as all keys installed on your machine.

On MacOS or UNIX platforms, use the **stshowkeys** command line tool to list the host identifier and license keys installed on the machine. This tool is in **$ROSE_BIN**.

```
% stshowkeys
CPU identifier for this machine:
   Host type ..... Sun SPARC/Solaris
   Host ident .... 7234799f

License Information ----------------
No licenses found
```

When you fill in the key request form, you will also need the host identifier of your machine (an 8-digit hex number) as well as the 6-digit STEP Tools license number that we assigned when you purchased your software.

Your request will be handled by our support staff and a key will be sent via email. When you receive your key, install it as described in the sections below. If you have any problems, please contact us at **license@steptools.com**.

## 7.2 Installing Keys on Windows

On Windows, license keys are kept in the registry. The key will arrive via email as a "**.stkey**" attachment. Save the file and double-click on it to load it into the registry. You can also run the ST-License Key Browser and select the **Load Key File** button.

Since the **stkey** files are just registry settings, you could also load them on your machine by running **regedit** and loading the key file using the **File | Import**, or by changing the file extension to "**.reg**" and double clicking the file.

After you install the license, run the ST-License Key Browser or the **stshowkeys** tool to list all STEP Tools product license keys. Save your key file in case you ever need to re-install.

## 7.3 Installing Keys on MacOS/UNIX

On MacOS and UNIX systems, the license keys for all STEP Tools products installed on your system are normally kept in the file **$ROSE/license**. If your system administrator prefers to rename or keep this file elsewhere, set the **$ROSE_LICENSE** environment variable to indicate the new location of the file:

```
setenv ROSE_LICENSE   /usr/local/common/step_keys
```

Your key will be sent to you in the body of an email message. To install the key, use a text editor cut and paste the license key text into this file.

```
# Sample STEP Tools license file
# License keys are series of hex digits beginning with a $.
# Lines beginning with a hash are ignored.
#
$e2c6081885 e2030f93cc 0bcd8919e7 e3c68c120a a3dc8c296b d4a2b8354a
 d4f4bf2c02 d4ffb57e51 cef5bd7e49 82f2e82900 80ebb82042 d7d2cc176f
 e382ed6e53 86e6df707e 82a4e36c18 8dafef730f
```

After you have added the license, you may verify the license with the **stshowkeys** tool. With no arguments, the **stshowkeys** program looks for an installed key file. It can also be given a filename argument to check the contents of a specific file.

```
% stshowkeys
CPU identifier for this machine:
   Host type ..... Sun SPARC/Solaris
   Host ident .... 7234799f

License Information ----------------
User name ............ Dave Shabotnick
User organization .... Foobar International
License serial ....... 17d4-7234799f-31fa4d1c-484
Licensed package ..... ST-Developer v9.0
License type ......... Single CPU/Node-locked
Host CPU type ........ Sun SPARC
Host identifier ...... 7234799f
Usable on this host: YES
```

# 8 Product Support

## 8.1 Overview

If you have questions regarding the installation or use of ST-Developer or any other STEP Tools product, please contact us via electronic mail at:

```
support@steptools.com
```

In addition, you may wish to visit the STEP Tools Web Site. This site has support files, current patches, tutorials, demos, and other helpful material.

```
http://www.steptools.com
http://www.steptools.com/support/
```

Finally, if you need to reach us by more traditional means, we can be contacted at:

```
STEP Tools, Inc.
14 First Street
Troy, New York 12180

(518) 687-2848
(518) 687-4420 (fax)
```

**STEP Tools, Inc**
**14 First Street**
**Troy, New York 12180**

**info@steptools.com**
**http://www.steptools.com**