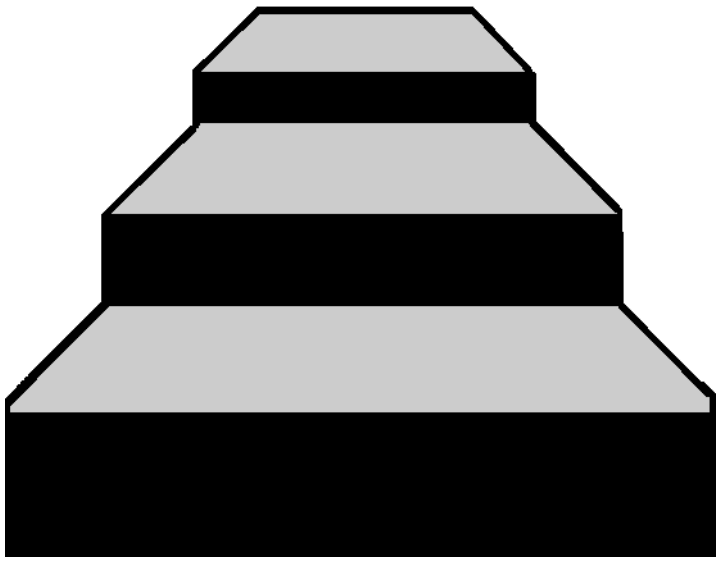
ST-Developer™ Tools Reference Manual



STEP Tools® Software

Contributors

This document grew from the combined efforts of many individuals. In particular, we would like to acknowledge the following people for their contributions: Thomas DeWeese, Matt Dinmore, Blair Downie, Jochen Fritz, Martin Hardwick, Dan Jacobs, Rick Kramer, Alexey Lipatov, David Loffredo, Alok Mehta, Brian Lloyd-Newbury, Jeff Young, and Chris Willis.

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 Introduction

1.1 What is STEP and ST-Developer?

The Standard for the Exchange of Product Model Data (STEP) is a family of ISO international standards that describe how to represent and exchange digital product information. In 1983, STEP merged earlier national efforts such as IGES, VDAFS, and SET. The first parts of STEP were published in 1994, including the EXPRESS information modeling language and the first application protocol, AP203, for exchanging BREP CAD geometry. Additional parts followed to address new technolgies, like Part 28 XML exchange (ISO 10303-28), and engineering information such as machine-tool control with STEP-NC AP238 (ISO 10303-238).

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.

1.1.1 The STEP Standard

Digital product data must contain enough information to cover a product's entire life cycle, from design to analysis, manufacture, quality control testing, inspection and product support functions. In order to do this, STEP must cover geometry, topology, tolerances, relationships, attributes, assemblies, configuration and more.

To accomplish this ambitious goal, STEP is a multi-part ISO standard. Many parts

are complete and published, while more are under development. These parts cover technology, such as testing procedures, file formats and APIs, as well as content, such as geometry, features, PDM, and other industry-specific information. The key aspect of STEP is extensibility. STEP is built on a formal language that describes the structure and correctness conditions of any engineering information that needs to be exchanged.

Industry experts use this language, called EXPRESS, to detail the information required to describe products of that industry. These "Application Protocols" form the bulk of the standard, and are the basis for STEP product data exchange. In addition, the EXPRESS language can document constraints as well as data structures. These formal constraints are an explicit correctness standard for the digital product data.

Figure 1.1 shows the structure of the STEP standard. Infrastructure parts, such as the exchange file format (Part 21), are separate from industry-specific application protocols, such as the drafting AP (Part 202, also called AP202). The industry-specific portion is open-ended. APs are available for mechanical design, drafting, analysis, process planning, CNC machining, shipbuilding, architecture and building construction, with more APs in development.

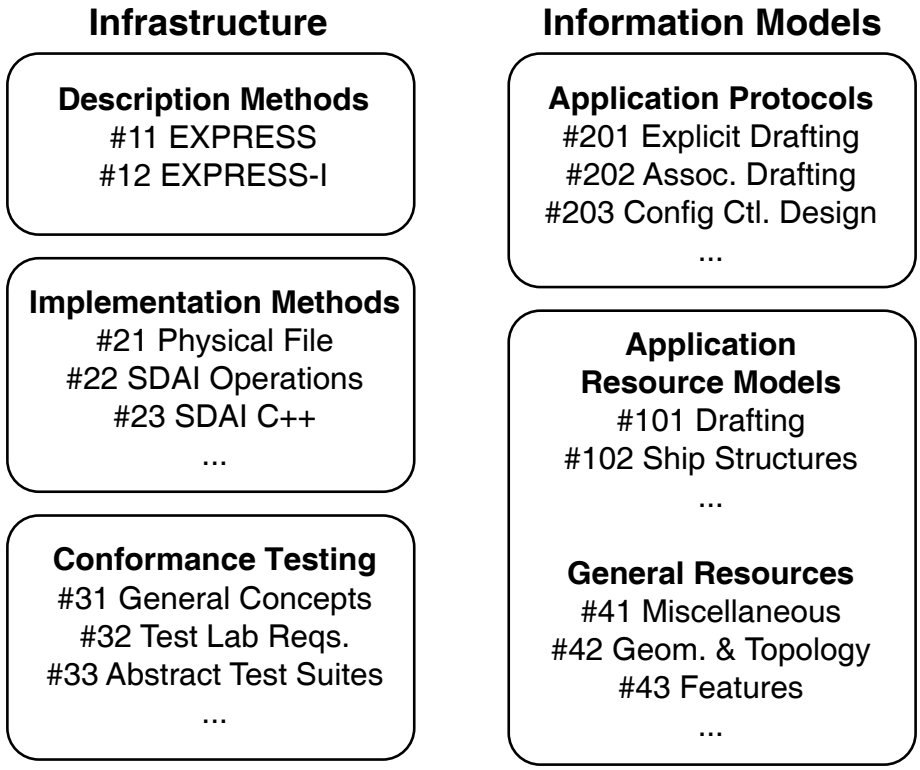


Figure 1.1 — Selected Parts of the STEP Standard

Some key reasons why STEP is important:

- STEP is a standard that can grow. It is based on a language (EXPRESS) and can be extended to any industry. A standard that grows will not be outdated as soon as it is published.

- The EXPRESS language describes constraints as well as data structure. Formal correctness rules will prevent conflicting interpretations. STEP SDKs use these descriptions to create more robust, maintainable systems.

- STEP is international, and was developed by users, not vendors. User-driven standards are results-oriented, while vendor-driven standards are technology-oriented. STEP has, and will continue to, survive changes in technology and can be used for long-term archiving of product data.

1.1.2 ST-Developer

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.

ST-Developer also includes a set of related tools, to perform various tasks on STEP data and EXPRESS definitions. This manual documents the following tools:

- **EXPRESS Compiler** — Produce C++ definitions from EXPRESS information models, then use these classes with the ROSE library to build application software or object-oriented databases.

- **EXPRESS-G Tools** — Construct EXPRESS-G diagrams from any EXPRESS information model, such as those in STEP, display, rearrange, and print them using a graphical editor.

- **STEP Conformance Testing Tools** — Check STEP data sets for adherence to the STEP Application Protocols using these tools. These systematically evaluate EXPRESS constraints and derived attributes to verify that a data set is correct.

- **Source Management Tools** — Manage and extend C++ class files, create makefiles, and perform other software development tasks.

- **IGES and AUTO-CAD DXF Converters** — Use these file format converters to convert data from systems that produce IGES or DXF files into STEP physical exchange files and vice-versa.

1.2 Getting Started

The *ST-Developer Release Notes* contain setup instructions for each platform. In general, you must make sure that the ST-Developer tools can find support files. On Windows systems, the support files are installed when ST-Developer is installed.

ST-Developer for Windows includes command-line and dialog versions of each tool described in this manual. It also includes the ST-Developer Launcher, a quick starting point for running tools. It is available on the Start menu and has icons for each tool, as well as a link to the ST-Developer online manuals. A shortcut to each tool is also available in the “ST-Developer Tools” folder on the Start Menu.

You can drag and drop files into the Windows dialogs or onto the icons in the ST-Developer Launcher, each one keeps a list or recently used files, and you can cut and paste, save or print the messages printed by the tools.

..

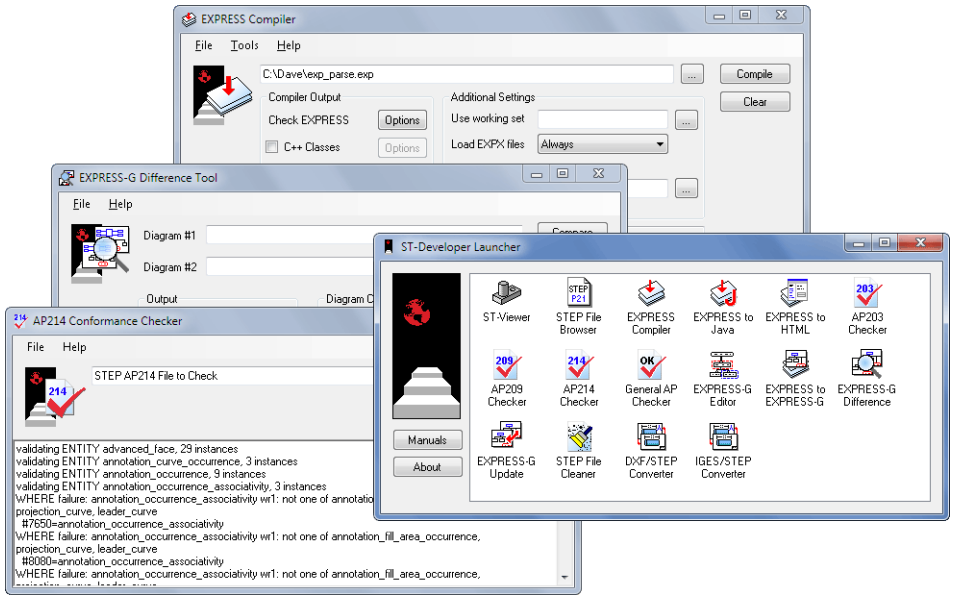


Figure 1.2 — ST-Developer Launcher with several tools.

On UNIX platforms, you tell the ST-Developer tools where to find support files using a special “logicals” file. This is normally run as part of your shell login scripts to set the variables.

On UNIX platforms, there are separate versions of this file for different shells. The **Rose_Logicals** file sets variables using the **csh** syntax, while the **Rose_Logicals.sh** file sets them using Bourne/Korn syntax. You should use them as follows. Replace **/usr/local/steptools** with the name of the ST-Developer installation directory:

```
% source /usr/local/steptools/Rose_Logicals  # CSH users
% set path=($path $ROSE_BIN)                 # set path

$ . /usr/local/steptools/Rose_Logicals.sh    # Bourne/Korn users
$ PATH=$PATH:$ROSE_BIN                       # set path
```

1.3 Manual Organization

This manual describes the executable tools included with ST-Developer. Consult the *ROSE Library Reference Manual*, *SDAI C Library Reference Manual* and *ST-Developer Java Library Reference Manual* for more information on the ST-Developer programming language bindings.

This manual is divided into the following sections:

- **Part One: EXPRESS Tools**
- **Part Two: EXPRESS-G Tools**
- **Part Three: Conformance and Editing Tools**
- **Part Four: Data Management and Development Tools**

Each section describes the command line and GUI tools, with examples, screen shots, and a complete listing of all options.

1.3.1 Font Conventions

Within the body of a paragraph, tools, functions, keywords, and filenames are denoted with bold san-serif font, such as **librose.a** or **/usr/local/bin**. Function names are shown with trailing parenthesis, such as **findDomain()** or **name()**. When referring to a C++ member function, the C++ scope notation is used, such as **RoseDesign::save()** or **RoseObject::domain()**.

We describe environment variable settings using either UNIX convention (**$VAR**) or the DOS/Windows convention (**%VAR%**)

Arguments for command-line tools are shown in a fixed-width font as below. The tool name is shown in bold. Optional arguments are shown in square brackets. An ellipsis (. . .) indicates that an argument can be repeated as needed.

```
expfront [options] expfile1 [expfile2 ...]
```

Function prototypes are listed with each parameter on a separate line and the function name set off in bold as shown below. Optional parameters are listed in standard C++ notation showing the default value.

```
RoseAggregate * findObjects(
            RoseAggregate * list_to_fill,
            RoseDesign* design = <Current>
            );
```

Occasionally, functions must be described in a parameterized manner, with angle brackets indicating the parameterized portion of the function definition. For example, the following prototype is used for the **RoseObject getInteger()**, **getString()**, **getObject()**, etc. functions:

```
<type_ref> get<name>(
            RoseAttribute * att
            );
```

Programming examples are shown in a separate paragraph, in a fixed-width font:

```
/* Create a point using the default constructor
 *  and use the update methods to set its values.
 */
Point * point1 = pnew Point;
point1->x (1.0);
point1->y (0.0);

/* Create points using a different constructor,
 *  that fills in the values in one step. */
Point * point2 = pnew Point (2.5, 4.0);
Point * point3 = pnew Point (5.0, 0.0);
```

Sample terminal sessions are shown in the same format, and the command prompt is indicated with a percent sign (%).

```
% expfront -classes geometry_schema.exp
% cd classes
% mkmakefile geometry
% make
```

1.3.2 References

The ISO STEP standards documents are the ultimate reference for the EXPRESS language and the application protocols. These documents should be available from your national standards body or directly from ISO.

The following publications may be of interest to readers looking for additional information on the STEP standard, or the EXPRESS information modeling language:

- J. Owen, *STEP: An Introduction*, Information Geometers (47 Stockers Avenue, Winchester SO22 5LB, United Kingdom), 1993.
- D. Schenck and P. Wilson, *Information Modeling the EXPRESS Way*, Oxford University Press, New York, 1994.

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 at:

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

This site has support files, tutorials, demos, sources for various documents, and other helpful material.

Part One: EXPRESS Tools

2 EXPRESS Compiler

2.1 Description

The EXPRESS compiler parses and checks information models defined in the EXPRESS language, then generates output that you can use with your programs.

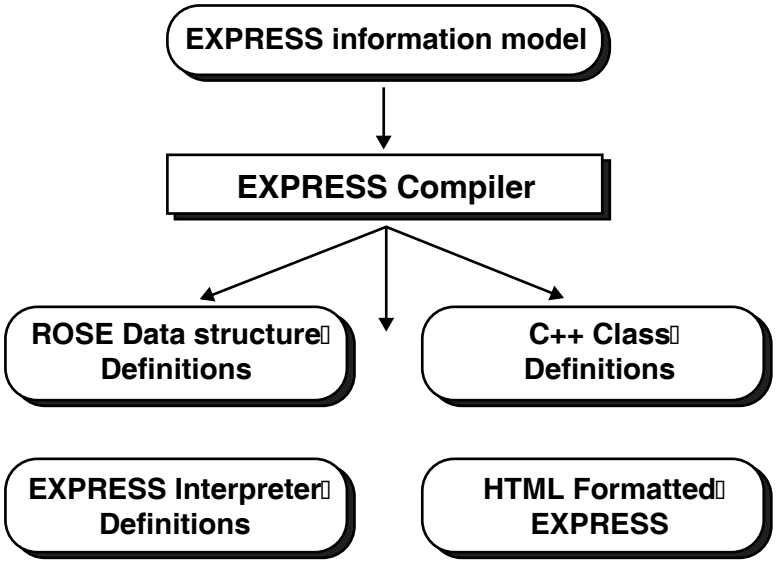


Figure 2.1 — EXPRESS Compilation

An EXPRESS information model contains data structure definitions along with matching rules and constraints. The compiler can do extensive parse and type

checking on such a model, and can produce:

- C++ class code for building programs around these structures.
- Java class code for building programs around these structures, using the **EXPRESS to Java Converter**.
- A data dictionary, in the ROSE data format, that application programs use to read and write STEP Part 21 files. These files are named after the schemas as **<schemaname>.rose**.
- EXPRESS parse information for the SDAI EXPRESS interpreter and AP conformance checking tool. These files are in the ROSE format, and are named after the schema with the **_EXPX** extension, as **<schemaname>_EXPX.rose**
- HTML and other formatted text from the EXPRESS definitions, using the **EXPRESS to HTML Converter** (Chapter 3, pp. 27).

The EXPRESS compiler can also compile “short-form” schemas and link them against libraries of resource schemas, such as the STEP 40-series schemas.

2.2 Command Line

expfront [options] expfile1 [expfile2 ...]

express2cxx [options] expfile1 [expfile2 ...]

The **express2cxx** wrapper calls **expfront** with the **-rose -classes** options.

If more than one EXPRESS file is given, the tool parses them as if all files were concatenated into a single file. If a filename has no extension, the tool appends “**.exp**” The compiler recognizes the following options.

-help Print the list of available options.

Options that control the error checking and warnings:

-level <level>
Set the level of checking, where **<level>** is one of **0**, **1**, **2**, **3**, **3+** or **3++**. Refer to **EXPRESS Schema Checking** (Section 2.5, pp. 18) for details. Default level is **3++**.

-local Suppress errors and warnings due to schemas that are not available

for checking.

-[no|very]strict
Control the strictness of schema checking. Strict checking is the default. Use **-nostrict** to produce fewer warnings. Refer to **EXPRESS Schema Checking** (Section 2.5, pp. 18) for details.

The **-nostrict** option will report only EXPRESS errors; no warnings will be reported. The **-strict** option excludes warnings that are excessively pedantic. The **-verystrict** option reports all warnings and errors.

-suppress Suppress errors and warnings on precompiled schemas.

Options that control the writing of compiled schema data and the loading of precompiled schema data:

-noload Prevents loading of precompiled schemas.

-nopath Prevents loading of precompiled schemas from the ST-Runtime support directories. Only precompiled schemas found in the current directory will be loaded.

-write[all|none]
The **-writeall** option forces writing of precompiled data for all schemas. The **-writenone** option prevents any precompiled data from being written.

-[no]write <schema>
Forces writing of precompiled data for given schema. The **-nowrite** option inhibits writing of precompiled data for given schema.

Options that control the generated classes and of compiled schema data and the loading of precompiled schema data:

-c Parse and resolve the EXPRESS schemas. No classes or ROSE data structure definitions are generated. This is the default behavior for the **expfront** tool. Equivalent to **-noclasses -norose**

-[no]classes
Generate / Do not generate C++ classes for the EXPRESS definitions. Default is to generate (**-classes**).

-[no]rose Generate / Do not generate ROSE data dictionary. The files are named after the original EXPRESS schemas **<schema-name>.rose**. Default is to generate (**-rose**).

-forcerose Generate a ROSE data dictionary even if one already exists.

The following options customize various aspects of the generated output. They are used in conjunction with the generate options above.

-aggfiles / -aggwithbase
Specify where the generated code for aggregates is placed. The default is **-aggwithbase**, which puts an aggregate into the same **.h** and **.cxx** files as its base type. The **-aggfiles** option puts each aggregate in a separate file.

-attcheck [local|global|none]
Specify how to check for conflicts with attribute names. The default is **local**, which checks for attributes that have the same name as their type. The **global** style checks for attribute names that conflict with any type in the schema. The **none** style disables all checks.

If a conflict is found, the compiler will prepend **att**_ to the attribute name in any generated code.

-cext <ext>
Specify a different body file extension for the generated class files. The default extension is **.cxx**. See **-hext** for changing the header file extension.

-[no]derived
Generate / Do not generate methods for derived attributes in C++ classes. The tool can generate access functions for derived attributes that do not contain function calls. As implemented, this may produce an invalid reference when one derived attribute depends upon another that was too complex to be generated. Default is not to generate (**-noderived**).

-entity <name>
Only generate C++ code for the given definition. This option assumes that you have previously generated code for any other referenced classes. It can be specified multiple times.

-hext <ext>
Specify a different header file extension for the generated class files. The default extension is **.h**. See **-cext** for changing the body file extension.

-[no]inverse
Generate / Do not generate methods for inverse attributes in C++ classes. The default behavior is not to generate such methods (**-noinverse**).

-namestyle [rose|sdai|fullsdai]
Name convention for generated C++ classes. The **rose** keyword selects the ROSE naming conventions; **sdai** selects SDAI C++ names without schema prefix; **fullsdai** selects full SDAI C++ names prefixed with schemas. The default is **rose**.

-o <dir> Destination directory for generated files. The directory is created if needed. If the directory exists, files will be moved to **<dir>.bak**. The default directory is **classes**.

-prefix <str> A name prefix for the generated C++ definitions. This will be prepended to all class and file names. A prefix can also be specified in a workingset file.

-schema <name>
Only generate C++ code for the given schema. This option assumes that you already have generated code for any other referenced classes. It can be specified multiple times.

-[no]virtual
Generate / Do Not Generate virtual base declarations in C++ classes. For code with no multiple inheritance, the **-novirtual** option will avoid the use of virtual base classes. Note, inappropriate use of this option can cause undefined behavior. Default is all superclasses virtual (**-virtual**).

-ws / -workingset <namefile>
Provide a control file with details about what to generate from the EXPRESS schemas. See **Customizing Output with a Working Set File** (Section 2.9, pp. 24) for more information.

Options for the shortform to longform converter

-shtolo Enable the short- to long- form converter.

-shortform <name>
The name of the schema to be converted to longform. The default value is the first schema in the first file.

-lfoutfile <file>
The output file for the generated longform schema. The default file is <schemaname>_longform.exp

Options that control other useful behavior:

-extension <ext>
Use input files ending with **<ext>** instead of **.exp**.

-force Forces the compiler to complete all passes, despite any errors.

-ignorefile <file>
Specify a file containing a list of types and entities that should be ignored.

2.3 Windows Control Panel

The EXPRESS Compiler Windows control panel is shown in Figure 2.2. Run this by selecting **EXPRESS Compiler** in the ST-Developer Launcher or the "ST-Developer Tools" Start Menu folder. The following sections describe the fields and setting on this control panel and show how to perform various tasks with the compiler.

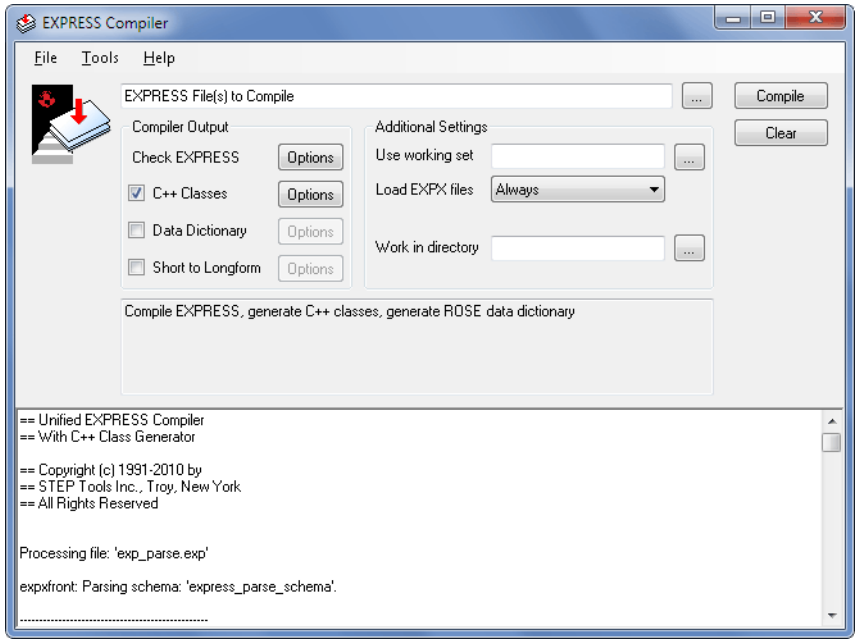


Figure 2.2 — EXPRESS Compiler Windows Control Panel

2.4 How the Compiler Works

The EXPRESS compiler reads a text file containing EXPRESS definitions, parses the definitions, and checks them for consistency. If multiple files are given, the tool behaves as if all files were concatenated together into a single file.

At the top of the EXPRESS Compiler control panel shown in Figure 2.2 is a text field for one or more schema files that you would like to compile. You can open the file dialog using **Ctrl+O** or the **[...]** button to the right. You can also drag and drop files from the Windows Explorer.

The checkboxes in the **Compiler Output** area control what sorts of things the compiler will generate from the EXPRESS definitions. Next to each checkbox is a button that brings up a dialog with additional options for that type of output.

The compiler always parses EXPRESS definitions, and checks them for consistency. From the command line, simply specify the EXPRESS file. No other arguments are required.

```
% expfront schema.exp
```

You can specify various options to vary the intensity of the checking and reporting. These are described in greater detail in **EXPRESS Schema Checking** (Section 2.5, pp. 18).

When **C++ Classes** is checked (as it is by default), or **-classes** is given on the command line, the compiler will write a C++ class definition for each entity, select, aggregate or enumeration in the EXPRESS file. By default the files are written to a subdirectory called **classes** under the working directory. See **Generate C++ Classes** (Section 2.6, pp. 20) for more information.

When **Data Dictionary** is checked or **-rose** is given on the command line, a file of compiled data structure definitions is generated for each schema. This data dictionary is used by application programs. Each file is put in the working directory and is named after the schema, **<schema>.rose**.

The compiler can also write parse information used by the SDAI EXPRESS interpreter and AP conformance checking tools to validate rules and functions. These are named after the schema with the **_EXPX** suffix: **<schema>_EXPX.rose**. These files are written into the working directory. See **Generate Data Dictionary and Precompiled Schemas** (Section 2.7, pp. 22) for more information.

When **Short to Longform** is checked or **-shtolo** is given on the command line, the compiler will expand a shortform schema (which references other schemas) into a longform schema (which is self-contained). See **Shortform to Longform Converter** (Section 2.10, pp. 24) for more information.

For more control over the generated output, you can specify a control file in the **Use working set** field. On the command line, you would use the **-ws** option. These files let you select subsets of definitions or other settings. See **Customizing Output with a Working Set File** (Section 2.9, pp. 24) for details.

The **Load EXPX** options are used with short form schemas. The compiler can resolve references by searching through precompiled definitions. See **Short Forms and Libraries of Precompiled Schemas** (Section 2.8, pp. 23) for details.

The **Work in directory** field controls where any generated files will be written. If you leave this blank, output will be put in the same directory as the first EXPRESS input file. When calling the compiler from the command line, all outputs are written to the current directory.

2.5 EXPRESS Schema Checking

The EXPRESS compiler parses EXPRESS definitions and performs extensive schema checking. The options described below vary the intensity of the checking and reporting. In the control panel, click on the **Check EXPRESS** options button to bring up a dialog with the options for parsing and resolving EXPRESS definitions..

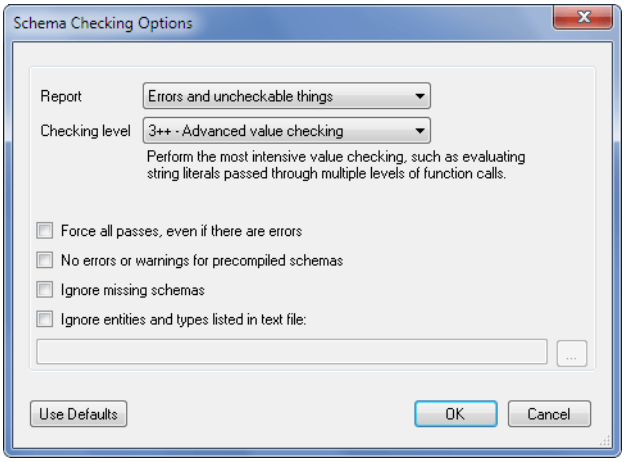


Figure 2.3 — Check EXPRESS Options Dialog

The compiler can check usage of the EXPRESS built-in functions **TYPEOF()**, **USEDIN()**, and **ROLESOF()**. Since parameters of these functions are often string literals, typographic and formatting errors are common. These flaws in the string literals are reported as warnings rather than errors by the compiler

Use the **Checking level** field or **-level** option to control the amount of checking performed by **expfront**. By default the compiler does all checking (**3++**). The compiler supports the following, and each level includes all checking done in lower levels.

`-level 0` **Syntax Checking** — Verifies that the input conforms to EXPRESS

language syntax. A few other types of errors will be detected, such as declaring an identifier more than once in a given scope.

`-level 1` **Structure and Reference Checking** — Examine the inheritance and type structure, and resolve all identifiers to the definitions they reference. Verify that the inheritance and definition graphs of entities, defined types and selects are acyclic.

`-level 2` **Type Checking** — Check that the result of expressions satisfy the constraints of type compatibility and assignment compatibility.

`-level 3` **Simple Value Checking** — Verify simple constraints on the values in the EXPRESS schema, such as whether aggregate lower bounds are less than upper bounds. More sophisticated constraints are checked by the following levels.

`-level 3+` **Intermediate Value Checking** — Check parameters to the built-in functions **USEDIN()**, **TYPEOF()**, and **ROLESOF()** for proper string formatting, resolution to defined types, and type compatibility.

`-level 3++` **Advanced Value Checking** — Perform the most intensive value checking. For example, the compiler will attempt to evaluate string literals passed through multiple levels of function calls.

These levels correspond roughly to the levels of checking defined in the EXPRESS Language Reference Manual (ISO 10303-11). However, this does not mean that the compiler performs all of the checks required by the standard for a particular level.

The **Report** field or the **-strict** and **-nostrict** options control how results are reported. With strict checking (the default), the compiler warns when it cannot completely check an EXPRESS construct. The construct may not be in error, but the compiler is unable to determine this. For example, when checking string values passed to **USEDIN()**, the compiler will warn if the string is constructed by an expressions more complicated than it can compute at compile time, so they can be checked by hand. Use the **-nostrict** option to suppress these warnings.

If you know that your schema has problems, you can select the **Force all passes** check box to make the compiler continue beyond the point where it would normally stop. On the command line, this is controlled by the **-force** option.

If you are linking against precompiled schemas that have errors, you can select **No errors or warnings for precompiled schemas** or by the **-suppress** option to avoid excessive messages associated with the precompiled schemas.

If you are compiling a short form schema and some schemas are not present, you can select **Ignore missing schemas** to avoid excessive errors associated with the

missing definitions. On the command line, this is controlled by the **-local** option.

Finally, functions sometimes refer to types that are not included in a schema. For example, the AP203 **valid_units** function (originally from Part 41) references several types that were not included in AP203, such as **luminous_intensity_measure**. This results in warnings like the one shown below:

```
"203.exp", line 5556: warning: Undefined entity or defined type
'TIME_MEASURE' in TYPEOF.
```

These warnings can be suppressed by specifying a file containing the names of types which should be assumed to exist. Use the **Ignore entities and tyles listed in text file field** or the **-ignorefile** option to provide the name of the file. The contents of the AP203 ignore file are shown below:

```
TIME_MEASURE
ELECTRIC_CURRENT_MEASURE
THERMODYNAMIC_TEMPERATURE_MEASURE
AMOUNT_OF_SUBSTANCE_MEASURE
LUMINOUS_INTENSITY_MEASURE
RATIO_MEASURE
SHELL_BASED_WIREFRAME_REPRESENTATION
GEOMETRICALLY_BOUNDED_WIREFRAME_REPRESENTATION
```

2.6 Generate C++ Classes

The compiler can generate C++ for use in building programs around the entity, select, aggregate and enumeration definitions in a schema. In the control panel, check the **C++ Classes** box or use the **-classes** option. The **C++ Classes Options** button brings up additional controls for customizing the generated code.

The C++ source code is written to **.cxx** and **.h** files in a subdirectory called **classes** under the working directory. If this would overwrite an existing file, the existing file is moved to a backup directory **classes.bak**.

The compiler generates a master include file **<schema>.h** that brings in all definitions for that schema. It also generates a **<schema>_ROSE_LOAD.h** file containing **ROSE_LOAD()** calls for each type in the schema. This can be used to force linkers to bring in all of the object files for a schema.

On the command line, the simplest way to generate classes is as follows. Programs that use the classes also need the data dictionary files generated by the **-rose** option. You can omit this if the data dictionary files have been previously generated, such as with the ST-Developer pre-installed schemas in **$ROSE/system_db/schemas**.

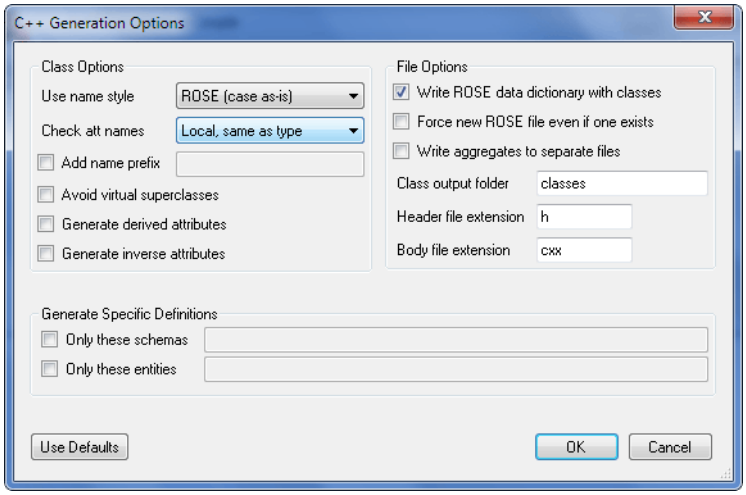


Figure 2.4 — C++ Classes Options Dialog

```
% expfront -classes -rose schema.exp
```

If you have a control file for selecting certain definitions or forcing AND/OR classes to be generated, specify it under the **Use working set** field. See **Customizing Output with a Working Set File** (Section 2.9, pp. 24) for more details.

In the **Class Options** section of the options dialog, the **Use name style** control or **-namestyle** command line option control how EXPRESS definition names are turned into C++ symbols. The default is to use the **rose** naming style, which uses the name with the same case as it appears in the EXPRESS. The **sdai** conventions change the case and possibly add a prefix. These are described in the *ROSE Library Reference Manual*.

The **Check att names** field and **-attcheck** option controls what happens when attributes and types use the same name. The default **local** style looks for attributes with the same name as their type. The **global** style checks attribute names for conflict with any type in the schema. The **none** style disables checking. If a conflict is found, the compiler will prepend **att_** to the attribute name in any generated code.

The **Add name prefix** field or **-prefix** option prepend a string to the definition names. This can also be done within a workingset file.

The **Avoid virtual superclasses** and **-novirtual** option. This will minimize the use of virtual base classes, but is not recommended because of potential conflicts if you add AND/OR or other multiple inheritance classes later on.

The **Use derived attributes** and **-derived** option attempt to generate code for some derived attributes. Only simple expressions can be handled in this way.

The **Use inverse attributes** and **-inverse** option attempt to generate code for inverse

attributes. Backpointers must be computed using the **rose_compute_backpointers** for the inverse attributes to work with acceptable performance, and it is up to the application programmer to make sure that they are current. Refer to the ROSE Library Reference Manual for more information.

In the **File Options** section of the options dialog, use the **Write ROSE data dictionary for classes** or **-rose** command line option to control the data dictionary files as described above. The **Force new ROSE file even if one exists** and **-forcerose** options will ignore any pre-installed data dictionaries and always write a new one.

Aggregates are normally put into the same **.h** and **.cxx** files as their base type. If you prefer separate source files for each aggregate, select **Write aggregates to separate files** or use the **-aggfiles** command line option.

By default, the generated classes are put in a subdirectory called **classes**. You can specify a different name in the **Classes output folder** field or use the **-o** option on the command line.

The generated C++ source files have a **.cxx** extension and the header files have a **.h** extension. If you prefer a different extension, you can specify it in the **Header file extension** field or the **Body file extension** field. On the command line, these are controlled by the **-cext** and **-hext** options.

The **Generate specific definitions** fields command the compiler to only produce C++ code for the given list of schemas or entities. This option assumes that you already have generated code for any other referenced classes. On the command line, this is controlled by the **-schema** and **-entity** options. You can specify them multiple times.

2.7 Generate Data Dictionary and Precompiled Schemas

The compiler can generate two types of schema dictionary information. Data dictionary files, **<schema>.rose**, are used by all application programs. The parse data files, **<schema>_EXPX.rose**, are used by the SDAI, AP conformance checker, and the compiler as described in **Short Forms and Libraries of Precompiled Schemas** (Section 2.8, pp. 23).

In the control panel, check the **Data Dictionary** box and use the associated **Options** button to bring up the options dialog. On the command line, use the **-rose** option to generate **<schema>.rose** data dictionary files. The **<schema>_EXPX.rose** parse data files are written by default when the command line version of the compiler is

used. To suppress them, use the **-nowrite** option:

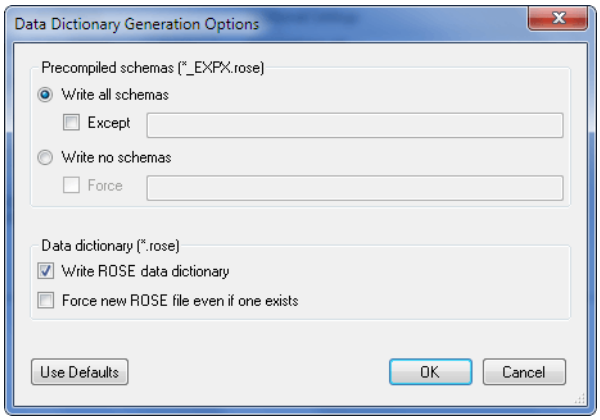


Figure 2.5 — Data Dictionary Options Dialog

```
% expfront -rose schema.exp
```

The **Precompiled schemas** options control which schemas should have precompiled parse information. The **Write all schemas** option and **-writeall** flag saves parse data for all input schemas. Use **Except** or the **-nowrite** command line option to suppress the writing of certain schemas.

The **Write no schemas** option and **-writenone** flag suppresses the writing of all parse data. Use **Force** or the **-write** command line option to selectively force the writing of certain schemas.

The **Write ROSE data dictionary for classes** or **-rose** command line option generates the data dictionary files. The **Force new ROSE file even if one exists** and **-forcerose** options will ignore any pre-installed data dictionaries and always write a new one.

Once you generate the data dictionary or precompiled parse data files you should consider copying them in to ST-Runtime installed schemas area so that other applications can find them.

2.8 Short Forms and Libraries of Precompiled Schemas

The compiler parses and resolves all of the EXPRESS definitions in the input files. If there are any unresolved schemas, the compiler will search for precompiled schema information to resolve those definitions. This allows, for example, the entire li-

brary of STEP Integrated Resources to be precompiled and automatically loaded whenever a STEP short-form AP is compiled.

This can be suppressed with the **-noload** option. The compiler searches for precompiled schemas in the ST-Runtime support directories. This behavior can be changed using the **-nopath** option.

Once all schemas have been either loaded or determined to be unavailable, the compiler checks the EXPRESS model as a whole. During this phase, the **-suppress** and **-local** options can control whether or not errors and warnings that are due to schemas that are either unavailable or that were precompiled and loaded are reported.

After checking is finished, precompiled parse data for each schema is written to a **<schema-name>_EXPX.rose** file. The files are always written to the current directory. Only schemas given as input files or precompiled schemas that changed (by being more completely resolved) are written. This behavior can be changed using the **-writeall**, **-writenone**, **-write**, or **-nowrite** options.

2.9 Customizing Output with a Working Set File

Working set files are small text file that give more information to the compiler about how you want your classes and other output generated. You can use these control files to generate a subset of definitions, change the naming of particular classes, or force the compiler to generate complex AND/OR classes or extra aggregates.

Tell the compiler to use a working set file with the **-ws** command line option or the **Use working set** field of the EXPRESS compiler control panel. Refer to Chapter 2 of the *ROSE Library Reference Manual* for a full discussion of working file syntax, usage, and options.

2.10 Shortform to Longform Converter

When **Short to Longform** is checked on the control panel or **-shtolo** is given on the command line, the compiler will expand a shortform schema (which references other schemas) into a longform schema (which is self-contained). The associated options dialog controls aspects of the process. .

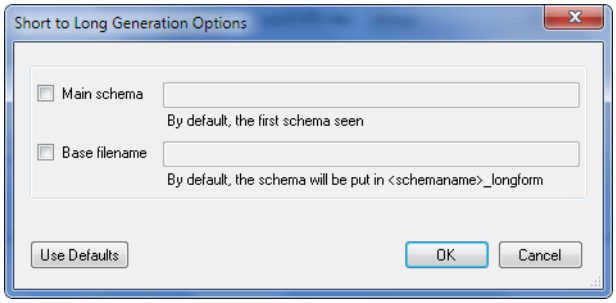


Figure 2.6 — Short to Longform Options Dialog

The **Main schema** option and the **-shortform** command line option give the name of the EXPRESS schema to be expanded. By default, the first schema in the first file will be expanded. Any remaining schemas are compiled and used to resolve REFERENCE FROM or USE FROM statements, superseding any separately-compiled schemas which may be available.

The **Base filename** option and the **-lfoutfile** command line option give the output filename for the longform schema. By default, the filename will be constructed from the schema as **<schemaname>_longform**.

3 EXPRESS to HTML Converter

3.1 Description

The **express2html** converter creates a web-browsable version of an EXPRESS information model. The tool preserves the layout, indentation and comments in the EXPRESS while adding hypertext links and cross reference information. This is not just a text formatter, but rather generates a report for each entity in the information model giving a more complete picture of the inheritance and attributes than is available explicitly in the EXPRESS.

In normal operation, the **express2html** tool reads the EXPRESS source files and generates one HTML page for each EXPRESS definition in a schema (entity, type, function, procedure, etc.) as well as a frame page that ties everything together. Each definition has a table of links to other definitions that use it, and ENTITY types also have tables listing all attributes, (both inherited and local) supertypes and subtypes.

3.2 Command Line

express2html [options] expfile1 [expfile2 ...]

-help Display usage information

-version Display the version number of the tool.

-title <text>
Set the title of the generated HTML to the specified string.

-exit <url> Set the target URL for the exit link. An exit link is only created if this option is specified.

-exittext <text>
Specify the text to be the exit link. The default is **[Exit]**. This option is ignored if the **-exit** option is not specified.

-terse Eliminate the sentences describing the sections from the HTML.

-hide Eliminate empty section from the report. For example if an entity has no supertypes, and this option is selected. the **Supertypes** section will not appear in the output for that entity.

-nojs Eliminate the JavaScript from the output. By default, JavaScript is emitted to enable the tables to be sorted by column.

-maxsubtypes <number>
Specify the maximum number of subtypes to include as a vertical list. If an entity has more subtypes than this number, they are listed horizontally to save space. The default value is 10.

-nocredit Do not include ST-Developer credits on the summary page.

-onefile Generate a single output file. In this case, the output consists of only the HTML marked-up EXPRESS, and does not include any summary information.

-frames Generate multiple output files with summary information. (This is the default operation of the tool.)

-o <path> Set the output location. If **-onefile** is in effect, this option specifies the HTML file where the output is written. If not specified, the output will be written to **express.html**

If **-frames** is in effect, this option specified a directory into which the output files are written. If not specified, the output will be written to the **express_html** directory.

-truncate When creating frames, limit files names to 10 characters (excluding the **.html** file extension). This is helpful when generating HTML for publication on CD-ROM or other file systems.

-comment-before
Comments appear before the definition to which they apply. (This is the default)

-comment-after
Comments appear after the definition to which they apply.

-comment-global
Used to indicate that comments in the global scope do not apply to either the preceding or the following definition.

3.3 Windows Control Panel

The EXPRESS to HTML Windows control panel is shown in Figure 3.1. Run this by selecting **EXPRESS to HTML** in the ST-Developer Launcher or the "ST-Developer Tools" Start Menu folder. The following sections describe the fields and setting on this control panel and show how to perform various tasks with the converter.

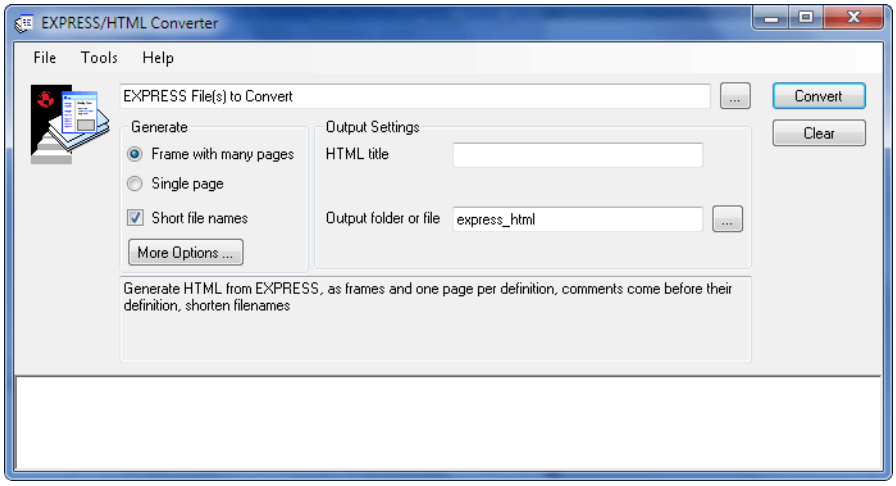


Figure 3.1 — EXPRESS to HTML Windows Control Panel

3.4 Generating HTML Output

At the top of the EXPRESS to HTML control panel shown in Figure 3.1 is a text field for one or more schema files that you would like to compile. You can open the file dialog using **Ctrl+O** or the **[...]** button to the right. You can also drag and drop files from the Windows Explorer.

The converter generates an web page for each entity, defined type, or other defini-

tion. It also generates index pages and a frameset to simplify navigation. The left panel provides an alphabetized index of either entities, defined types, functions or procedures. The right frame contains the formatted definition. The bottom frame is used to select an index to be displayed in the left frame. The **[all definitions]** tag in the bottom frame selects the summary page.

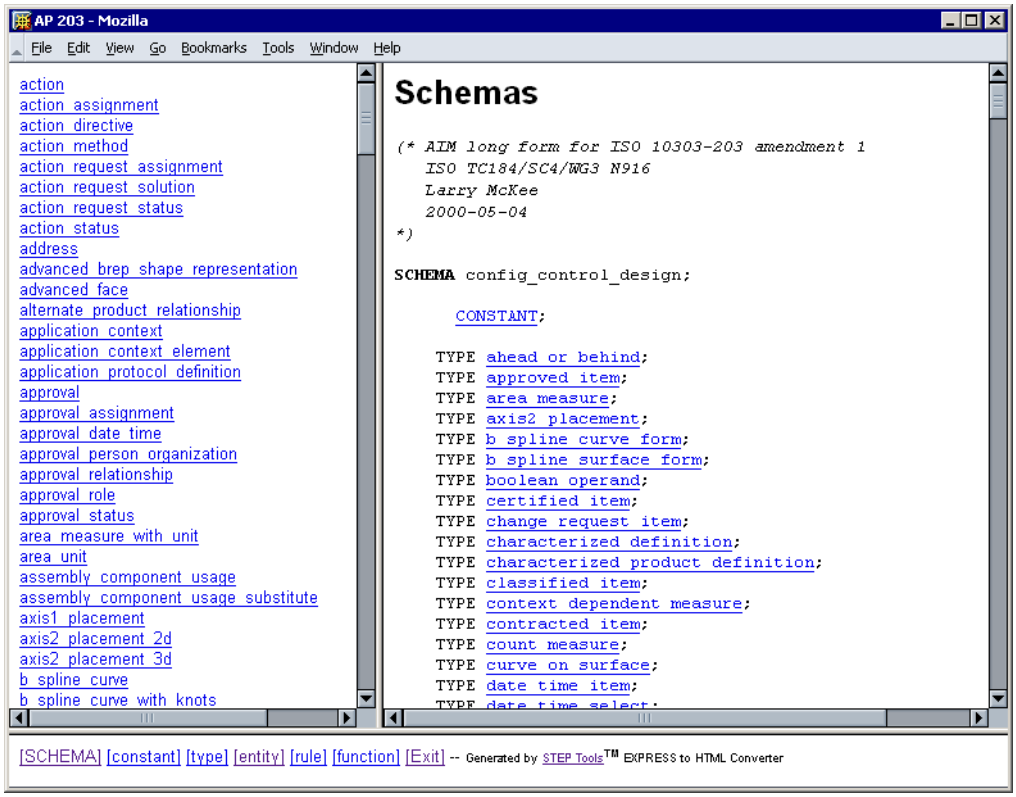


Figure 3.2 — HTML Output Showing Summary Page

In the output directory, the frameset is written into a file named **index.html**. You should point your web browser at this file when you want to view the hypertext version of the information model.

The summary page **schema.html** contains the HTML for the global information in the EXPRESS source. This includes the **SCHEMA**/**END_SCHEMA** declaration, **CONSTANT**s, **USE** and **REFERENCE** statements, and comments that appear in the global scope that the processor did not include on an HTML page for a definition. In place of those definitions the summary page includes a hypertext link to the page containing the definition.

The processor places all entities, defined types, global rules, subroutines and functions on their own HTML pages. In addition to the hyperlinked EXPRESS source, the pages for entity and defined types include a cross reference report at the bottom of the page. This report includes attribute summaries, subtypes and supertypes, and

the types and entities that reference the entity.

The converter generates tables for each entity to summarize the entity's attributes. For each entity, the tool generates summaries for explicit, derived, and inverse attributes. These tables give the attribute name, the attribute's type and the entity (possibly a supertype) that defined the attribute. The explicit attributes in the table are listed in the same order that they appear in a Part 21 exchange file.

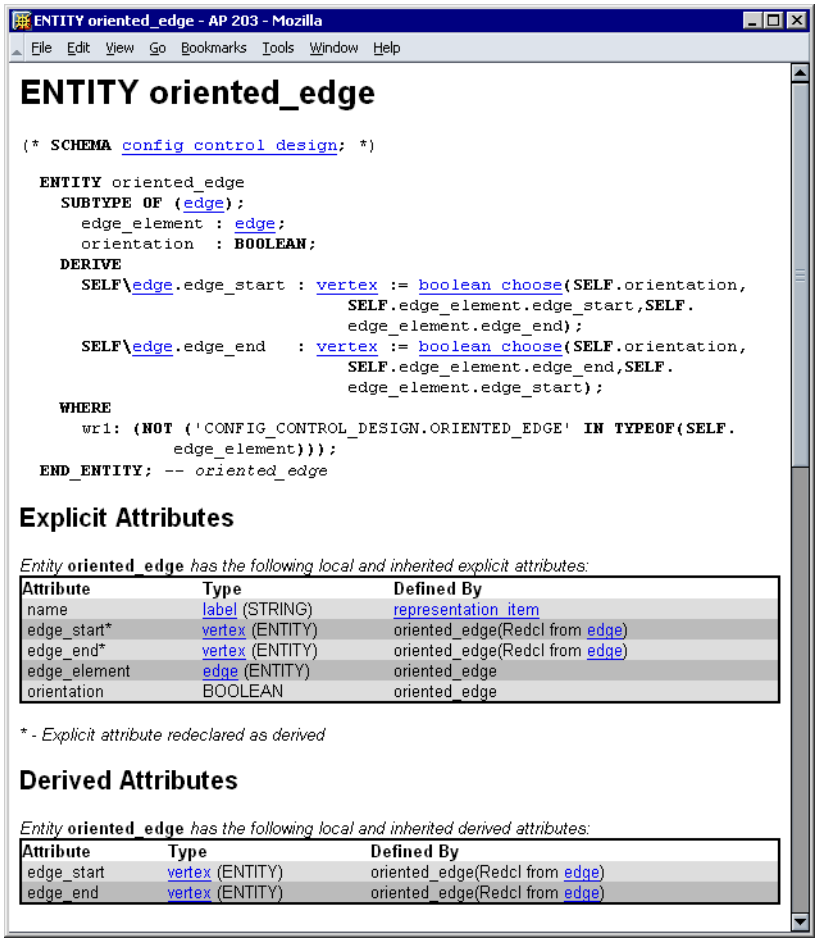


Figure 3.3 — Attribute Information for an Entity

The EXPRESS language allows explicit attribute to be redeclared as DERIVED. In this case, the attribute, even though it is a derived attribute, it is also included in the table of explicit attributes. It is indicated as such with a * and a footnote. In Part 21 files these attributes are encoded as "*".

The second column of the attribute table identifies the attribute's type. The type is specified as the EXPRESS type that it is actually declared as, and if that type is a named type, the underlying type (e.g. BOOLEAN, STRING, ENTITY, ARRAY, etc.), is also given in parenthesis. This feature makes it easy to determine the underlying type of defined types.

Both entity and defined type pages include a report identifying the other types that reference the current type. This section allows you to see where in the information model an entity or type is used, and effectively provides a back link for the hypertext links in the formatted EXPRESS.

3.5 Customizing the HTML Output

The converter has a range of options for customizing the output web pages. The checkboxes in the **Generate** area control how the EXPRESS definitions will be converted to web pages.

The **Frame with many pages** option generates detailed reports for each definition. The **Single page** option or **-onefile** command line flag tags all of the definitions with hyperlinks but does not generate any other reports.

You can specify title text for the web pages using the **HTML title** field or the **-title** command line option. The **More Options** button brings up an options dialog with even more settings.

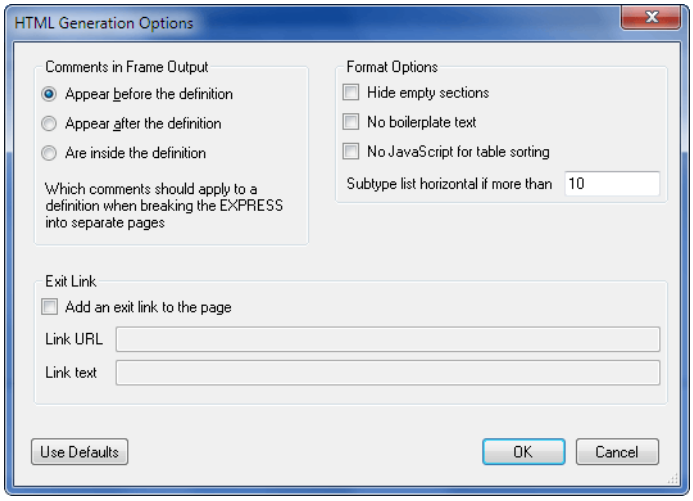


Figure 3.4 — EXPRESS to HTML Options Dialog

When generating multiple files, the tool needs to determine where one definition ends and another begins. Such splitting can be performed anywhere between the end of the preceding definition and the beginning of the next one. The splitting strategy must handle comments that lexically occur between definitions, but actually document either the following, or the preceding definition in the information model.

The **Comments in Frame Output** options determine where the converter will break

the EXPRESS into separate files. You can indicate that definitions are described by comments that appear before it, after it, or neither. Given the following information model:

```
SCHEMA flying_monkeys;

(*
The census entity  is used to hold a count of how many flying monkeys have
passed a point, and the time at which the last one was observed.
*)
ENTITY census;
  monkey_count : integer;
  last : time;
END_ENTITY;  --census

(*
  The bite entity is used to keep track of when a flying monkey bites a
  person, and, whether or not the person was infected with rabies by the
  monkey.  If known, the ID tag from the collar of the monkey is also
  recorded.
*)
ENTITY bite;
  victim : STRING;
  infected : LOGICAL;  (*Logical because the status may not be known*)
  monkey : OPTIONAL id;
END_ENTITY;  --bite

END_SCHEMA;
```

There are comments in the schema scope that come before the definition they document. In this case, use the **Appear before the definition** or **-comment-before** option to keep the comments with the associated definition. This is the default behavior. Use the **Appear after the definition** or **-comment-after** option in cases where the comments appear after the declaration.

When we use the “before” option, the output for the **census** entity includes the comment that appears before it, as well as the trailing comment on the same line as the **END_ENTITY** keyword as shown below:

```
(*
The census entity  is used to hold a count of how many flying monkeys have
passed a point, and the time at which the last one was observed.
*)
ENTITY census;
  monkey_count : integer;
  last : time;
END_ENTITY;  --census
```

The EXPRESS is always split between lines, when possible. Thus trailing comments after the entity remain with the entity regardless of the splitting mode selected. In the above example, the trailing comment “**--census**” has been kept with the

definition it belongs to.

The **Are inside the definition** or **-comment-global** option puts comments that are outside of a definition onto the schema summary page, not with any definition. **Comment Splitting Example** (Section 3.6.1, pp. 35) shows a more detailed example of how each of these options work.

The **Hide empty sections** option and **-hide** command line flag supress section headers for things that are not present in a definition, like supertypes, subtypes, etc. The **No boilerplate text** option and **-terse** command line flag supress some extra descriptive text describing each section.

The generated lists of local and inherited attributes normally contains some JavaScript code that can sort the lists by different columns. The **No JavaScript for table sorting** option and **-nojs** command line flag supress this code and emit static tables.

The page for each entity lists all types derived from it, either directly or through intermediate entities. This list is easiest to read when formatted vertically, with one definition per line, but this list can get very large for base types of a large inheritance hierarchy. The **Subtype list horizontal if more than** option and **-maxsubtypes** command line flag give the size at which that list switches to a more compact horizontal layout.

The **Exit Link** option and **-exit**/**-exittext** command line flags add a link to the bottom of every page.

3.6 Examples

Generate a set of HTML pages for the AP203 schema. The output is placed into a directory named **express_html**:

```
% express2html ap203.exp
```

Generate a single HTML file. This will contain a the EXPRESS schema with hyperlinks for each symbol usage, but will not have the summary tables for each definition. The output is placed into a file named **express.html**:

```
% express2html -onefile ap203.exp
```

Create a set of pages in the directory **ap214** with the page titles set to **AP 214**. We must quote the title to insure that the whitespace is correctly interpreted.

```
% express2html -o ap214 -title "AP 214" ap214.exp
```

Generate a set of pages for the STEP integrated resources. The output is written to a directory named **IRs**.

```
% express2html -o IRs p41.exp p42.exp p42.exp p43.exp p44.exp p45.exp p46.exp
```

3.6.1 Comment Splitting Example

As an example of how the comment splitting options work, we will consider the following simple EXPRESS schema, and show how it gets split in each case.

```
(*A schema for demonstrating how splitting is done*)
SCHEMA test;
(* Comment at beginning*)

ENTITY A;
(*Comment inside A*)
END_ENTITY; --A

(*Comment between A and B*)

ENTITY B;
(*Comment inside B *)
END_ENTITY; --B

(*Comment at end*)
END_SCHEMA;
(*After the schema*)
```

With the **-comment-after** option, the summary file contains the following:

```
(*A schema for demonstrating how splitting is done*)
SCHEMA test;
(* Comment at beginning*)
ENTITY A;
ENTITY B;
END_SCHEMA;
(*After the schema*)
```

The page for ENTITY A contains the following text:

```
ENTITY A;
(*Comment inside A*)
END_ENTITY; --A

(*Comment between A and B*)
```

The page for ENTITY B contains the following text:

```
ENTITY B;
(*Comment inside B*)
END_ENTITY; --B

(*Comment at end*)
```

For the **-comment-global** option, the summary file contains the comments.

```
(*A schema for demonstrating how splitting is done*)
SCHEMA test;
(* Comment at beginning*)
ENTITY A;

(*Comment between A and B*)

ENTITY B;

(*Comment at end*)

END_SCHEMA;
(*After the schema*)
```

In this case, the HTML for entity A, contains the following:

```
ENTITY A;
(*Comment inside A*)
END_ENTITY; --A
```

Note that the trailing comment "**--A**" is still kept with the entity because it appears on the same line as the **END_ENTITY** keyword.

Part Two: EXPRESS-G Tools

4 EXPRESS-G Language Overview

4.1 Description

EXPRESS-G is a standard graphical notation for information models. It is a useful companion to the EXPRESS language for displaying entity and type definitions, relationships and cardinality. For information on the EXPRESS-G notation, consult Annex B of the EXPRESS Language Reference Manual (ISO 10303-11)

The EXPRESS information models within STEP application protocols can become quite complex. EXPRESS-G diagrams help you master this complexity. With the tools described below, you can convert EXPRESS text to EXPRESS-G diagrams, view and edit them with a visual editor, and print them on any PostScript® or HP-GL based printer/plotter. ST-Developer contains the following EXPRESS-G utilities:

EXPRESS-G Editor (Chapter 5, pp. 45) — Interactively display and edit EXPRESS-G diagrams.

EXPRESS-G Layout Engine (Chapter 6, pp. 81) — Convert EXPRESS text to EXPRESS-G diagrams.

EXPRESS-G Diagram Update (Chapter 7, pp. 89) — Preserve the layout of the EXPRESS-G diagram, when the underlying EXPRESS changes.

EXPRESS-G Diagram Compare (Chapter 8, pp. 93) — Highlight similarities and/or differences between different versions of an EXPRESS schema.

EXPRESS-G Diagram Information (Chapter 9, pp. 97) — Extract page size, defini-

tion index and other information from a diagram.

EXPRESS-G to PostScript Converter (Chapter 10, pp. 99) — Convert diagrams to PostScript for printing.

EXPRESS-G to HP-GL Converter (Chapter 11, pp. 101) — Convert diagrams to HP-GL for plotting.

EXPRESS-G Set Options (Chapter 12, pp. 103) — Set the page size, font name, and size, and orientation (useful for old EXPRESS-G documents).

4.2 Why EXPRESS-G?

EXPRESS-G diagrams are an aid for understanding large information models. The diagrams show relationships and structure more clearly than the plain EXPRESS text.

The **expgedit** visual editor provides interactive access to diagrams. The **expg2ps** tool generates output that can be printed on any PostScript compatible printer, and **expg2hpgl** generates output that can be plotted by any HP-GL based plotter.

All of these tools operate on EXPRESS-G diagrams generated by the **express2expg** tool. This tool produces diagrams from plain EXPRESS text. The diagram layout is usually good enough to be used without modification, so you can simply regenerate the EXPRESS-G diagrams when you want to view changes to the original EXPRESS text.

The **expgupdate** program is a powerful tool for dealing with changes in your schema. This tool will preserve your investment in arranging your EXPRESS-G document when the underlying EXPRESS text changes.

The **expgdiff** tool can highlight any similarities and/or differences between two EXPRESS-G schemas. In many ways it is similar to a UNIX diff that knows about the syntax of EXPRESS.

In some situations, you may wish to improve upon the default layout, such as by arranging objects onto a few pages for printing. Use the **expgedit** editor to arrange the definitions as needed. Using the light gray lines on the screen as page guides, move a box by clicking and holding down the first (left most) mouse button. If you double click with the first mouse button it will also select any attached boxes. It is also possible to drag out a selection box, and everything that touches the box will then be selected. The shift key in cooperation with the mouse allows you to modify

the current selection.

When you are done, save your changes using the "save" button. You can print your diagram from the editor on the Windows platform. On UNIX, you can use **expg2ps** to generate PostScript that can be piped to the UNIX **lpr** utility.

4.3 Examples

The following example uses the tools to view and print an EXPRESS information model. We will use the command line versions of the tools, although you may normally use them from the ST-Developer control panel (Windows) or ST-Workbench (UNIX). We will look at AP-203, one of the standard STEP information models. The STEP models can be found in the ST-Developer release directory, under the **$ROSE/express/part203** directory.

1. As with all of the ST-Developer tools, make sure the environment variable **$ROSE** is set to the ST-Developer install directory, and that **$ROSE_BIN** is in your search path. Consult the ST-Developer release notes for details.

2. Generate the EXPRESS-G diagrams from your EXPRESS text with the diagram layout tool:

```
% express2expg p203.exp
```

3. View the diagrams with the **expgedit** editor.

```
% expgedit p203.exg
```

4. If you would like a wall-mounted copy of the schema for your office, print the diagrams. Please note that the printout will be over 100 pages long. On Windows, you can print directly from **expgedit**. On UNIX, use one of the print filters. In this example, we will print to a PostScript printer using **expg2ps** and **lpr**. If you have a plotter, you may wish to use the **expg2hpgl** tool instead.

```
% expg2ps p203.exg | lpr
```

The following example shows how one would use these tools to compare two different versions of an EXPRESS text file, using **expgdiff**. In this case, we will look at university and univ-extend. The EXPRESS for “university,” and “univ-extend” can be found in the ST-Developer release directory, under **$ROSE/express/local**.

1. Generate EXPRESS-G diagrams using the layout engine.

```
% express2expg university.exp
% express2expg univ-extend.exp
```

2. You may want to view the diagrams with the **expgedit** editor at this point.

```
% expgedit university.exg univ-extend.exg
```

3. Compare the two EXPRESS-G documents using **expgdiff**. If you want to see what entities, and types the diff tool correlates you can add the **-verbose** option as well.

```
% expgdiff university.exg univ-extend.exg
```

4. Now view the diagrams again with the **expgedit** editor. You should now see that almost all of **university.exg** is highlighted, and all of the corresponding entities in **univ-extend** are also highlighted.

```
% expgedit university.exg univ-extend.exg
```

5. Now repeat using the STEP information models **p203.exp**, and **p42.exp**. These can be found in **$ROSE/express/part203** and **part42**.

The following example shows how to carry over changes from an older version of an EXPRESS file to a new version, using **expgupdate**. In this case, we will look at university and univ-extend. These EXPRESS schemas can be found in the release directory, under **$ROSE/express/local**.

1. Generate the EXPRESS-G diagrams from your EXPRESS text with the **express2expg** tool:

```
% express2expg university.exp
% express2expg univ-extend.exp
```

2. At this point you may wish to view the diagrams with **expgedit**.

```
% expgedit university.exg univ-extend.exg
```

3. Now we will update the university EXPRESS-G diagram with the univ-extend EXPRESS, using **expgupdate**. If you want to see what entities, and types the tool correlates you can add the **-verbose** option as well.

```
% expgupdate university.exg univ-extend.exg
```

4. View the diagrams again with the **expgedit** editor. The most noticeable change is that 'student' and 'course' have been moved up to where they were located in the university EXPRESS-G document.

```
% expgedit university.exg univ-extend.exg
```

The following example shows how to use the working set option of **express2expg** to make large EXPRESS schemas easier to understand. The working set option selects a subset of the schema for processing. This selection is made by a simple text file that contains the entities that you are interested in working with. These entities and all of the entities above them in the inheritance hierarchy are automatically pulled into the EXPRESS-G diagram. Any references to objects not in the current working set are replaced with inter-schema references.

1. Create a working set file for Part-42. Use any entities you wish, but two interesting ones are: **b_spline_surface_with_knots**, and **quasi_uniform_surface**. Create a file with the names of some entities you are interested in. Put each name on a separate line. Refer to **Customizing Output with a Working Set File** (Section 2.9, pp. 24) for more information about working set syntax.

2. Generate the EXPRESS-G diagram from your EXPRESS text using the working set file created above. Part-42 can be found in **$ROSE/express/part42**.

   ```
   % express2expg -workingset my_entity_names part-42.exp
   ```

3. Now view the diagram with the **expgedit** editor.

   ```
   % expgedit part-42.exg
   ```

5 EXPRESS-G Editor

5.1 Description

The **expgedit** tool is the EXPRESS-G diagram viewer and editor. This tool can display the diagrams produced by the **express2expg** tool. The tool allows you to scroll through the EXPRESS-G data model, follow page references, move groups of objects, and redirect lines between objects.

5.2 Command Line (All Platforms)

expgedit [options] [datafile]

The UNIX version of the EXPRESS-G editor recognizes the options shown below. In addition, the tool accepts standard X11 options.

-bg <color> Set the background color. The default is white.

-pg <color> Set the color of the page grids. The default is gray.

The UNIX version must be started on an X11 display running. A color display is recommended for best appearance. If you get a “Cannot open display” message, the tool has been unable to connect to your X server. Check the **$DISPLAY** environment variable or use the **-display** option. If necessary, use the **xhost** utility to enable access to your display server.

The Windows version of EXPRESS-G Editor accepts the options shown below:

/p <file>
Print the file to the default printer and exit.

/pt <file> <printer_driver> <port>
Print the file to the specified printer and exit. For example, if you had the Adobe Acrobat package installed on your machine, you could run the following to get a PDF version of your diagram:

```
> expgedit /pt diagram.exg "Acrobat PDFWriter"
```

To run EXPRESS-G Editor on the Windows platform:

1. On the taskbar, click the **Start** button.
2. Point to **Programs** and then point to the **ST-Developer** or **ST-EXPRESS** folder.
3. Click **EXPRESS-G Editor**.

Alternatively you can start the editor from the command line.

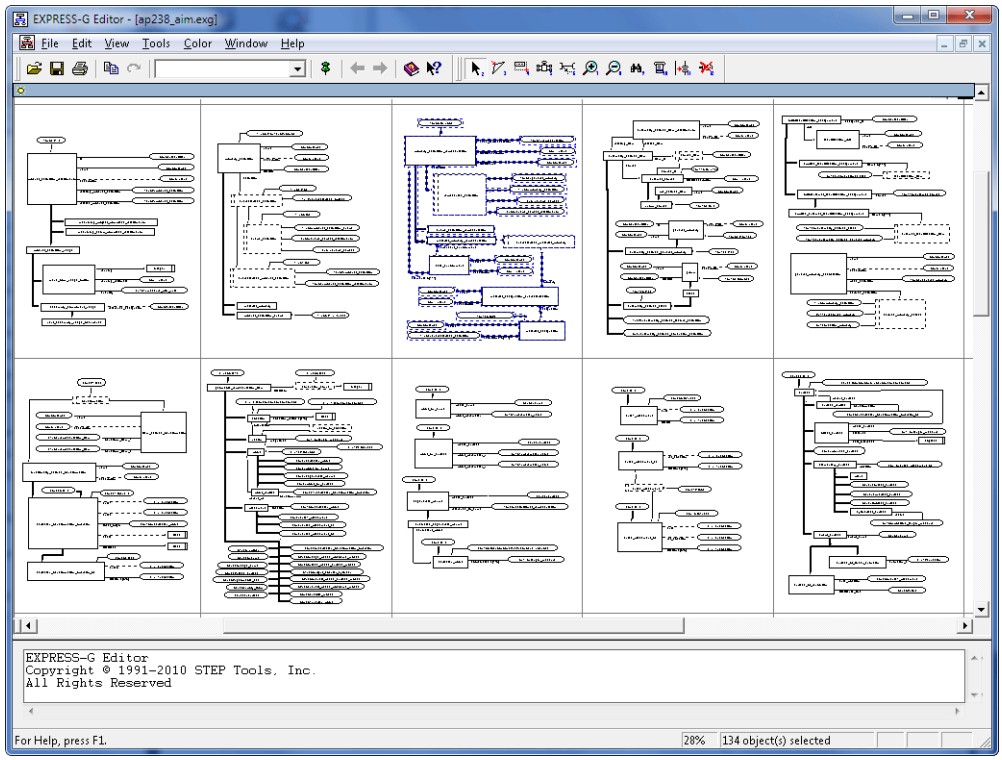


Figure 5.1 — EXPRESS-G Editor (Windows version)

5.3 Overview

EXPRESS-G Editor displays diagrams in a large scrolling workspace. Above the workspace are menus and a toolbar. Below the workspace is a status bar that gives the font name/size to use for printing, the page number of the upper right hand corner of the display, and the location of that page in the grid of pages.

The Windows version also contains the message window where information, warning, and error messages are printed. On UNIX systems these messages are printed in the terminal window where the editor was started.

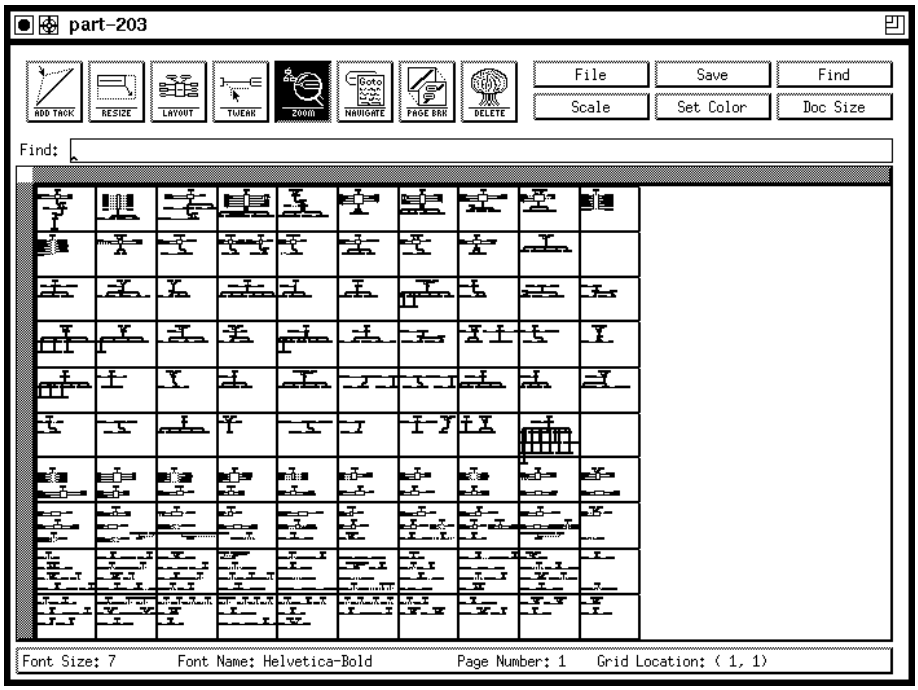


Figure 5.2 — EXPRESS-G Editor (UNIX version)

5.3.1 Changing Display Scale

Initially, the workspace is scaled so that you can see the entire diagram. To zoom in:

UNIX — Click the **Zoom** button on the toolbar. Once this tool is activated you can click in a spot with the *second* (middle) mouse button to zoom in. To zoom out, click again. You can also choose a scale from the **Scale** menu.

Windows — Either press **<F9>** or choose the **Toggle Zoom** tool on the **Tools** menu, or choose **Scale** on the **View** menu and set scale to 100% in the **Scale** dialog box. You can also use the **Quick Scale** menu accessible through the scale indicator located on the

status bar: 25% .

5.3.2 Moving through the Diagram

You can move through a diagram using scroll bars, the **Panner** window (Windows version only), the search box (described in the next section), or by following page references. Page references connect two definitions, such as a super/sub type relationship, without having a line physically connecting them. In addition, there are special methods for navigating between entities in the same inheritance branch and between entities connected by a reference path through attributes.

To use the page references and the special methods to travel around the document, you must first activate the Navigate tool. See **Toolbars** (Section 5.7, pp. 74).

UNIX — Click the **Navigate** button on the toolbar. Once this tool is activated you can click page references and entities with the *second* (middle) mouse button.

Windows — Activate the Navigation tool using the toolbar button , the **Tools | Navigate** command, or the **<F10>** key. Once the tool is activated you can click page references and entities with the *left* mouse button.

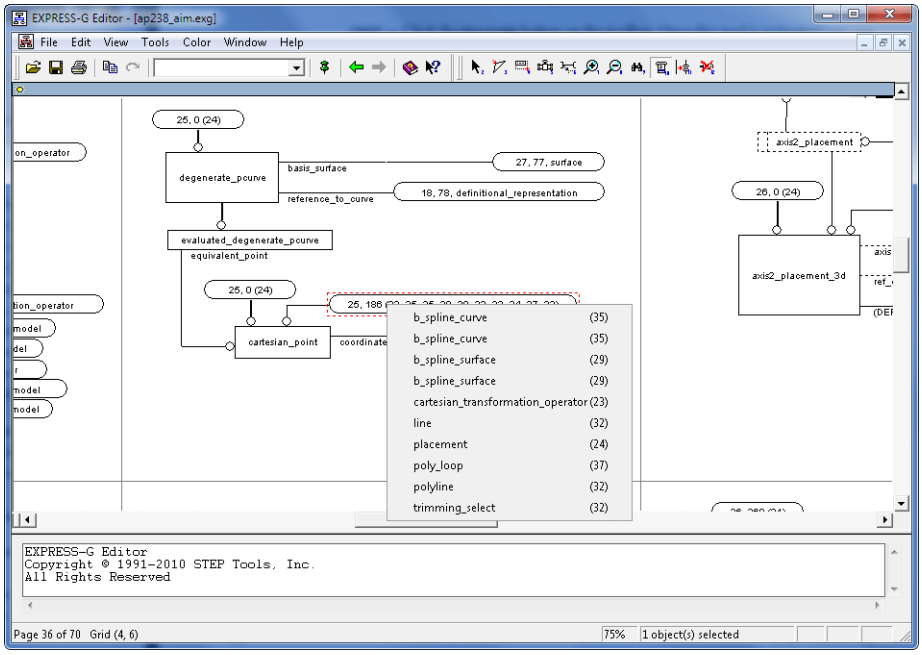


Figure 5.3 — Page Reference Menu

For outgoing page references, you will travel immediately to the other end of the

reference. For incoming page references a menu will appear where you can select which page reference you want to visit. This menu is shown in Figure 5.3.

For entities, you will travel through two levels of menus, first selecting the type of relationship (forward reference, backward reference, supertype, subtype), then selecting from the resulting list. Reference and inheritance lists are indented representations of a tree, where the greater indentation indicates a further distance in the selected direction. The punctuation in these lists corresponds to the punctuation used in STEP AP Mapping Tables.

Symbol	Meaning
.	Entity to one of its attributes
->	Attribute to its type
<-	Attribute type from attribute
=>	Supertype to subtype
<=	Subtype to supertype
=	Select to one of its components
1=>	Supertype to a ONEOF subtype
1<=	ONEOF subtype to supertype
(above)	Duplicate entity, see above

5.3.3 Navigation Sequence

Windows — The Editor stores the locations where the navigation was initiated, so that you can return to the previous location by using the **View | Back** command. These stored locations form a navigation sequence that can be traversed back and forth by using the **View | Back** and **View | Forward** commands. A navigation sequence is updated by the following operations:

- Page reference navigation
- Search (**Edit | Find** and **Edit | Find Next** commands)
- **Edit | Go To** command

5.3.4 Searching for Definitions

You can also find specific definitions by typing a name or regular expression in the search box at the top of the editor. Pressing **<Enter>** will scroll the diagram to the first matching definition. The matching definition will be surrounded by a red

bounding outline. To find the next match, press **<Enter>** again or use the **View | Find Next** command (Windows version only).

The search operaton uses regular expressions. It accepts file name completion syntax (i.e. **'R*'** will find any entities, defined types, etc. that have 'R' as their first letter, it is case insensitive). It also offers alternation **'[abcd]'**, and exclusion **'[^abcd]'**. If you want access to full regular expressions you may enclose the expression in parentheses (i.e. **'R.*'** means any entity that starts with 'R.' where as **'(R.*)'** means any entity that starts with just 'R').

The search can also find entities using Part 21 short names. In order for this method to work there must be a short name file somewhere in your rose search path. Short name files are identified by having the same name as the EXPRESS-G file only with the extension **'.nam'**.

5.3.5 Selecting Objects in the Workspace.

Selecting and moving objects is accomplished using the Select tool. When the Select tool is active and you click an object, that object will become selected and the previous selection will be cleared. To keep the old selection and add a new object to it, hold down **<Shift>** while selecting the new object. You can also remove objects from the selection by holding down **<Shift>** and clicking the objects you want to remove. To select a group of objects, click in the diagram blank space and drag out a box that surrounds all of the objects you want to select. Once you selected an object or objects you can drag the selection to a new location.

You can also select larger groups of definitions. Double-click an entity to select the entity and its attached attributes. If you hold down **<Ctrl>** while you double-click an entity, that box and all of its attached boxes, as well as all of the boxes below it in the inheritance tree will become selected. This is extremely useful for moving entire inheritance trees around.

Windows — When the active tool is other than Select, you can still temporarily use the Select tool by pressing and holding down the **<Spacebar>** key.

5.3.6 Mouse Operation

Left Mouse Button

UNIX — Always used for selections and dragging (see commands below).

Windows — Always performs whatever action the active tool is programmed to perform. See the tool descriptions for a more detailed information. When the active tool is **Select**, use the following operaitons.

Drag Make a rectangle, objects that are inside the rectangle will be selected.

Single-click Select the object.

Double-click Select the object as well as any attributes of that object, or edit the bookmark if it is the selected object.

<Ctrl>+Double-click
Select the object as well as any attributes of that object, and repeat this for all of the children of this object in the inheritance tree.

When the **<Shift>** key is held down while making a selection, all of the newly selected objects have their selection state reversed.

Middle Mouse Button

UNIX — This always performs whatever action the active tool is programmed to perform. See the tool descriptions below for a more detailed information.

Windows — If available, the middle button is used for panning. If you hold down the middle button and move the pointer, the diagram will move as if it were attached to the pointer.

Right Mouse Button

Windows — This button always displays the shortcut menu. See **Shortuct Menus** (Section 5.5.8, pp. 69) for more information on shortcut menus.

Wheel button (*IntelliMouse pointing device and compatible devices*)

Windows — When pressed, the wheel button works as the middle mouse button. When rotated the wheel button scrolls the diagram (vertically if **<Shift>** is released or horizontally if **<Shift>** is held down), and zooms the diagram if the **<Ctrl>** key is held down.

Restricted Movements

Windows — The restricted movements feature allows you to align and resize objects

more accurately. To use this feature, select an object, and then press and hold down the **<Alt>** key while moving the mouse. If you start moving the mouse horizontally, vertically or in any diagonal direction, the selected object will move only in this direction, regardless of how you continue to move the mouse. For example if you started to move the object in horizontal direction, the vertical coordinate of the object will remain the same while you hold the **<Alt>** key even if the mouse pointer moves in vertical direction.

5.4 The Workspace

The workspace is a scrolling work area containing an EXPRESS-G diagram. The diagram is marked with a light grey grid to indicate where the page boundaries will be when the file is printed. Figure 5.4 shows an example of this. A box or line that crosses a page separator will be split across multiple pages. You can reposition objects or insert page breaks as needed to prevent this.

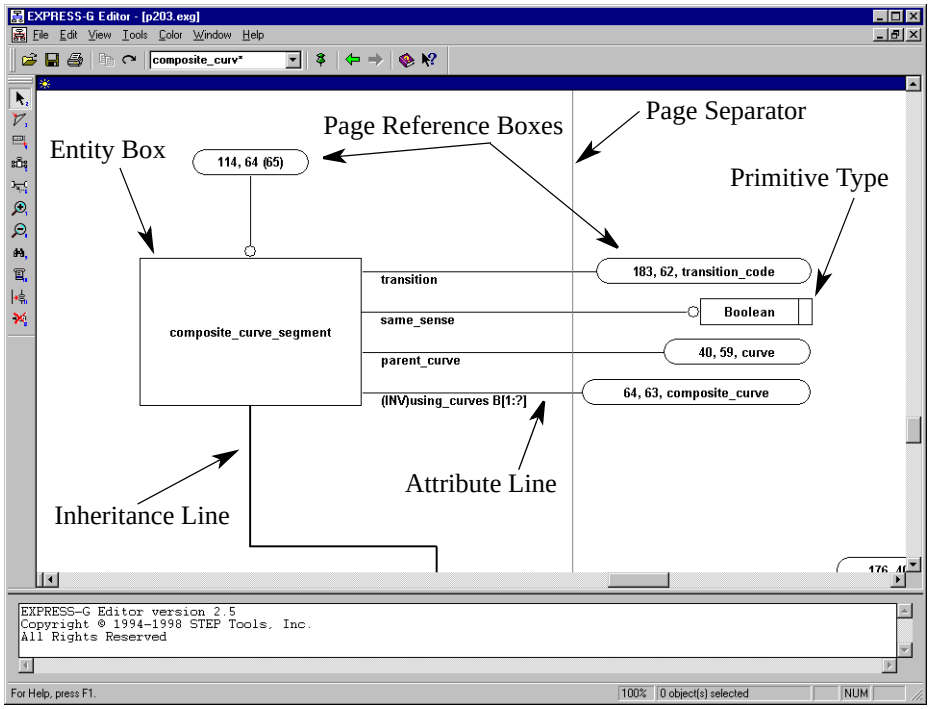


Figure 5.4 — The Major EXPRESS-G Objects

5.5 Menus (Windows Version)

Each of the sections below describes a menu and the commands which appear on it. Some of these commands are also available on one of the toolbars, or may have a keyboard shortcut. We have shown toolbar icons and keyboard shortcuts where appropriate

5.5.1 File Menu

Open — (Also **<Ctrl+O>**) Shows the **Open File** dialog box. Using this dialog box you can browse the file system and locate the file to open.

Close — Closes the active diagram window. If the diagram was modified, EXPRESS-G Editor will prompt whether to save the diagram or not.

Save — (Also **<Ctrl+S>**) Saves the active diagram in the file it was opened from.

Save As — displays the **Save As** dialog box. You can then select a file to which you want to save the active diagram or type a new file name.

Print — (Also **<Ctrl+P>**) displays the **Print** dialog box, where you may specify the range of pages to be printed, the number of copies, the destination printer, and other printer options.

When you have objects selected in the diagram, only these objects will be printed by default (the **Selection** option under **Print Range** is selected). Click **All** under **Print Range** if you want the whole diagram to be printed.

If the command is carried out by the toolbar button it does not display the **Print** dialog, but starts printing immediately.

Print Preview — Displays the diagram in the preview window exactly as it would be printed on paper (see Figure 5.5). This way you can check your layout before you actually start printing.

Print Setup — Use this command to select a printer and a printer connection. This command displays the **Print Setup** dialog box, where you specify the printer and its connection.

Page Setup — Use this command to specify the headers and footers on a page. This command displays the **Page Setup** dialog box (see Figure 5.6 on page 54).

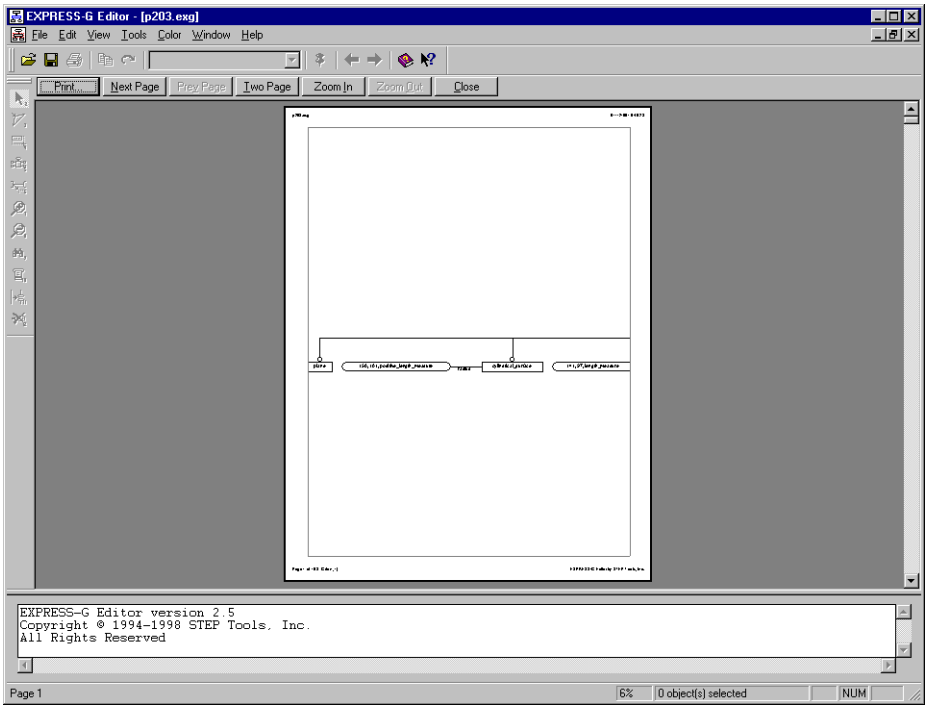


Figure 5.5 — Print Preview Window

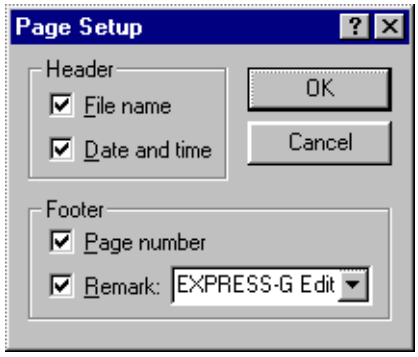


Figure 5.6 — Page Setup Dialog Box

This dialog has the following options for the header and footer:

- **File Name** — Select to print the file name in the upper-left corner of a page.
- **Date and time** — Select to print the time of printing in the upper-right corner of a page.
- **Page number** — Select to print the page number and grid location in the lower-left corner of a page.

- **Remark** — Print a remark (the product name by default) in the lower-right corner of a page. You can type the text of the remark in this box. It also keeps a history list of previously used remarks, to open the list click the button.

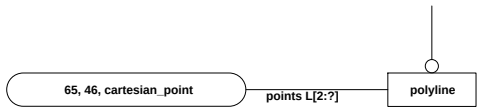
Send — Use this command to send the active diagram through e-mail. This command displays a message window of your default e-mail program with a blank e-mail message and the active diagram as an attachment. You can then use your e-mail program to send the message.

Recent files — These items correspond to the most recently opened files. Choose one of these to open the file immediately.

Exit — Quits EXPRESS-G Editor.

5.5.2 Edit Menu

Copy — (Also **<Ctrl+C>**) Copies the selected objects to the Clipboard using the Windows Metafile format. The objects can be pasted into any Windows application that supports this format. The example below was copied from one of the STEP APs.



Some publishing programs allow you to edit each graphical object in the pasted data, e.g. to set font, line style and thickness, shading, etc.

Save Selection As — Saves the selected portion of the diagram to a file for later import into a document. You can save the selection as a Windows Metafile (**.wmf**) or Encapsulated PostScript (**.eps**) file. The EPS produced by the EXPRESS-G Editor does not include TIFF preview.

Select All — (Also **<Ctrl+A>**) Selects all objects in the diagram. Selected objects are displayed with the blue bounding outline.

Find — (Also **<Ctrl+F>** and) This command is used to find objects in the diagram using regular expressions. When selected it displays the **Find**

dialog box (see Figure 5.7).

Figure 5.7 — Find Dialog Box

To find an object in the diagram type the regular expression in the **Object name** box or use the list to select one of the previously used. To open the list, click the button. For more information on regular expressions see **Searching for Definitions** (Section 5.3.4, pp. 49).

Find Next — (Also **<Ctrl+Right Arrow>**) Use this command to find the next box matching the current search pattern.

Go To — (Also **<Ctrl+G>**) This command displays the **Go To** dialog box (see Figure 5.8). Use this dialog box to specify a page or bookmark in the diagram and scroll the diagram instantly to display the specified location

.

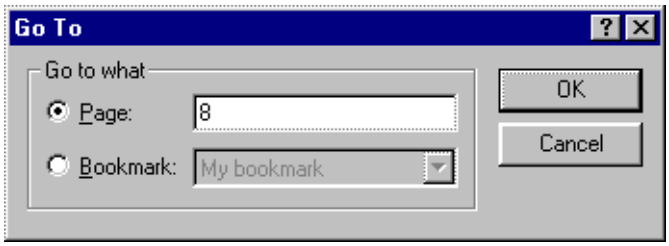


Figure 5.8 — Go To Dialog Box

Insert Bookmarks — Use this command to insert bookmarks in the diagram. When this command is chosen the pointer becomes a .

Click in the diagram to insert a new bookmark. EXPRESS-G Editor shows a text input box where you clicked. Type the bookmark text and press **<Enter>**. The bookmark will be displayed at the same place.

You can move the bookmark by selecting and dragging it to the new location when the active tool is Select. You can edit the bookmark text by right-clicking it and choosing the **Edit Bookmark** command on the **Object** shortcut menu. You can delete a bookmark by choosing **Delete Bookmark** on the **Object** shortcut menu.

The bookmarks can be displayed as pushpins and labels or as pushpins only. They

also can be hidden so that they are neither displayed nor printed. The **View | Bookmarks** menu commands control how the bookmarks are displayed.

Remove All Bookmarks — Removes all bookmarks from the diagram.

Add Row — Adds a new row of pages to the bottom of the diagram.

Insert Row — Inserts a new row of pages at the top of the diagram. All objects in the diagram are shifted down by one page.

Add Column — Adds a new column of pages to the right of the diagram.

Insert Column — inserts a new column of pages at the left side of the diagram. All objects in the diagram are shifted right by one page.

Remove Row — Removes the last row of pages. The pages in this row should not contain any objects, otherwise it will not be removed.

Remove Column — Removes the last column of pages. The pages in this column should not contain any objects, otherwise it will not be removed.

Properties — (Also **<Alt+Enter>**) This command displays the **Diagram Properties** dialog box shown in Figure 5.9. The following options are available:

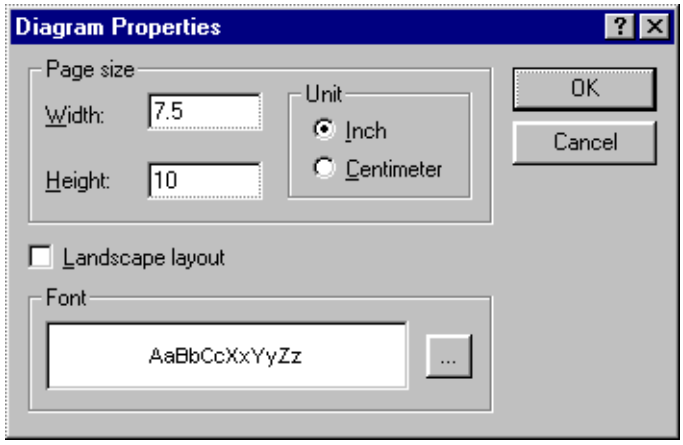


Figure 5.9 — Diagram Properties Dialog Box

- Page size and orientation. You can set the width and height of a page in either inches or centimeters. When you select the **Landscape layout** check box the width and height values will swap.

- Font specifications. Click the **[...]** button to display the **Font** dialog box that you can use to select any font installed on your system with any attributes, such as bold, italic, color, underline, and overstrike. These font specifications are not recognizable by the UNIX version of the editor — it will continue to use the

default font (Helvetica Bold).

5.5.3 View Menu

Commands in this menu control the appearance of both the main window and a diagram window.

Toolbars — Displays a menu with the following commands:

- **Standard** — Displays or hides the standard toolbar, which includes buttons for some of the most common commands in EXPRESS-G Editor. The standard toolbar is normally located across the top of the main window, under the menu bar.

- **Diagram Tools** — Displays or hides the **Diagram Tools** toolbar, which includes buttons for the tools from the **Tools** menu.

- **Large Buttons** — Enlarges the toolbar buttons so that they are easier to see. This command is useful when giving presentations or for those who have poor vision.

Status Bar — Displays or hides the status bar at the bottom of the main window. Status bar describes the action to be executed by the selected menu command or toolbar button, and the keyboard latch state. It also shows the current page number and its grid location when the pointer is over the diagram workspace, as well as the current scale and the number of currently selected objects.You can also set the scale using the **Quick Scale** menu which is displayed when you click on the scale indicator 

on the status bar.

Messages — Displays or hides the message window located at the bottom of the main window. Information, warning, and error messages appear in this window during the tool operation. You can also resize this window vertically by dragging its resize bar with the mouse.

All Diagram — (Also **<Ctrl+L>**) Scales the diagram to fit everything into the diagram window.

Full Size — (Also **<Ctrl+U>**) Displays the diagram in full size (scale of 100%).

Panner — (Also **<Ctrl+R>**) Displays or hides the **Panner** window shown in Figure 5.10. The **Panner** window is like a two-dimensional scrollbar. When you drag the floating box in the window, the diagram will scroll accordingly. The aspect ratio of the **Panner** window is the same as the aspect ratio of the diagram, while the aspect ratio of the floating box is that of the diagram window.

Figure 5.10 — Panner Window

Forward — (Also **<Alt+Right Arrow>**) Scrolls the diagram to the next location in the navigation sequence. If the end of the sequence is reached this command becomes unavailable. For more information on navigation sequences see Section 5.3.3 on page 49.

Back — (Also **<Alt+Left Arrow>**) Scrolls the diagram to the previous location in the navigation sequence. If the beginning of the sequence is reached this command becomes unavailable. For more information on navigation sequences see Section 5.3.3 on page 49.

Bookmarks — Displays a menu with the following commands:

- **Pushpins and Labels** — Displays full bookmarks (as pushpins and labels).
- **Pushpins Only** — Displays bookmarks only as pushpins (no text labels).
- **Hide All** — Hides all bookmarks without actually removing them.

Page Grid — Displays or hides the page grid. If the grid is displayed in the window, it will also be printed as a thin frame around each page. If the grid is hidden, the pages will be printed without frames.

Partial Model — This command controls the appearance of the primitive type boxes in the diagram. When this command is checked the primitive type (e.g. **REAL**, **LOGICAL**, etc.) boxes are hidden.

Scale — Displays the **Scale** dialog box shown in Figure 5.11. This dialog box has three sliders that control the current scale, the increment scale (used by the **Zoom In/ Out** tools) and the toggle scale (the second scale used by the **Toggle Zoom** tool).

The default values are:

- Current scale — set to fit the whole diagram.
- Increment scale — 20%
- Toggle scale — 100%

Working Set — Displays the Working Set dialog box (Figure 5.12), which you can

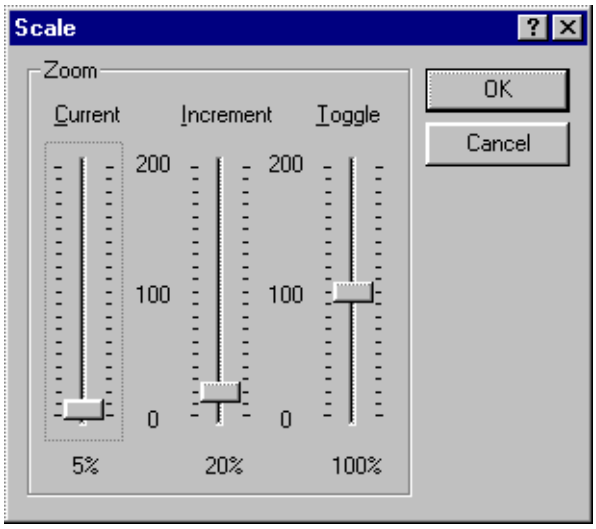


Figure 5.11 — Scale Dialog Box

use to view, edit, and apply the working set.

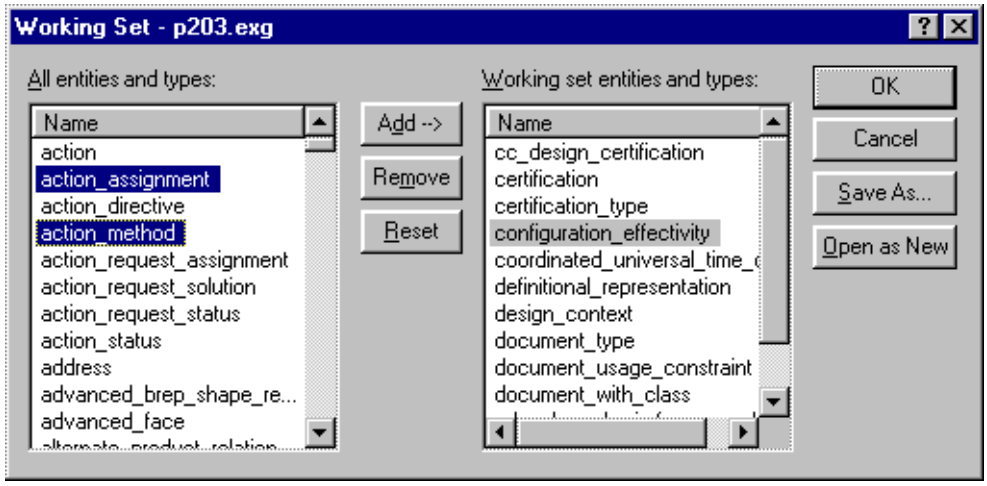


Figure 5.12 — Working Set Dialog Box

This dialog box has two lists that are used to manipulate the working set. The **All entities and types** list contains names of all entities in the current diagram. The **Working set entities and types** list contains the names of entities included in the working set. You can add entities to the working set by selecting them in the **All entities and types list** and dragging them to the **Working set entities and types** list or clicking the **Add** button. To remove entities from the working set select them in the **Working set entities and types** list and then click **Remove**. To remove all entities from the working set click **Reset**.

To save the working set in a file click the **Save As** button. This will display the **Save As** dialog box. You can save the working set either as a working set text file or as an EXPRESS-G diagram. If the working set is saved in the text file this file

can be later used by EXPRESS Compiler or the EXPRESS-G Layout tool. Click the **Open As New** button to generate a new EXPRESS-G diagram using the specified working set and then load this diagram into the editor. The new diagram name will be constructed from the name of the active diagram by adding the **_ws_#** suffix, where the # is a number 1, 2, etc. For example if the original diagram was named **MyDiagram.exg**, then the new diagram will be named **MyDiagram_ws_1.exg**.

Click **OK** to close the dialog box and apply the changes in the working set to the current diagram (all entities in the working set will be marked by the working set color). Click **Cancel** to close the dialog box and ignore the changes.

5.5.4 Tools Menu

The commands in this menu activate different tools used to edit the diagram.

Select — (Also **<F2>**) Activates the Select tool, which is used to select and move objects. When this tool is activated the pointer becomes an . For more information on selections and moving see Section 5.3.5 on page 50.

Add Tack — (Also **<F3>**) Activates the Add Tack tool. The Add Tack tool can be used to create new tacks on existing lines. When the tool is activated the mouse pointer becomes a .

To create a new tack click the line to which you wish to add a tack and drag the tack to the desired location.

Resize — (Also **<F4>**) Activates the Resize tool. The Resize tool is used to resize boxes. When this tool is activated the mouse pointer becomes a .

To resize a box click in the box and drag the border of the box to the desired size.

Layout — (Also **<F5>**) Activates the Layout tool. The Layout tool reformats the boxes that are connected to an entity. When this tool is activated the mouse pointer becomes an .

When you click an **ENTITY**, **ENUMERATION**, or defined type box the following menu appears:

- **All Attributes to the Left** — Lays out all attribute boxes on the left side of the entity/type box.

- **All Attrubutes to the Right** — Lays out all attribute boxes on the right side of the entity/type box.

- **Attributes Alternate Sides —** Lays out all attribute boxes evenly on the left and right of the entity/type box.

- **Custom —** Lays out each connected box separately using the **Custom Layout** dialog box.

When you click a **SELECT** type box the menu will have these additonal commands:

- **Fan-out Above** — Lays out the boxes of the select list in a fan-like fashion above the **SELECT** box.

- **Fan-out Below** — Lays out the boxes of the select list in a fan-like fashion below the **SELECT** box.

This tool can also be used to reestablish the Manhattan layout of lines. If you click a line, the following menu appears:

- **Lay out Entire Line** — Delete all tacks and apply the Manhattan layout.

- **Lay out Line Segment** — Apply the Manhattan layout to the selected line segment (between the two closest tacks).

If you click a forked tack, the **Lay out Entire Line** command will lay out the whole tree of lines.

Custom Layout Dialog Box

The **Custom Layout** dialog box is shown in Figure 5.13.

The **Custom Layout** dialog box has five lists — one that contains all connections of the selected entity/type box, and four lists for each side of the entity box. You can lay out connections by selecting them in the **Connections** list and then moving them to the side boxes. To move you can either click the corresponding button in the **Move** area or simply drag the selected item(s) into one of the side boxes. You can also drag items between side boxes.

The **Connections** list has three columns. The **Name** column contains the names of the connections (attribute name, page references or sub/supertype name). The **Type** column contains the types of connections (attribute type, page reference to/from, or sub/supertype relationship). The **Modifier** column contains the additional connection properties, such as **OPTIONAL**, **INVERSE** or **DERIVED** for attributes. To sort the list according to a column click the column header. To change the width of a column

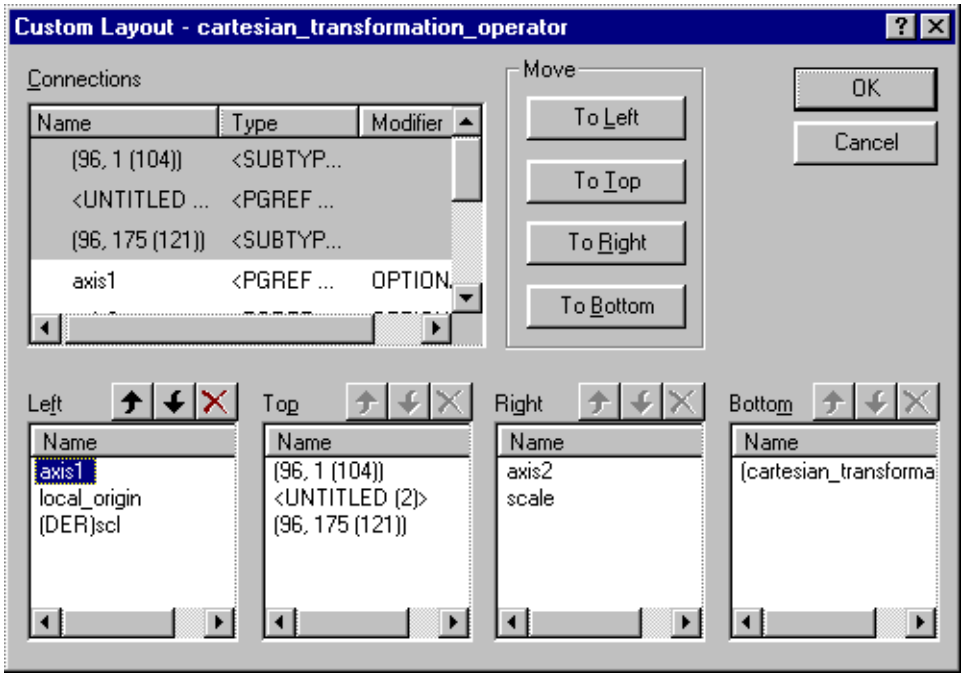


Figure 5.13 — Custom Layout Dialog Box

drag the column separator. To adjust the width automatically, double-click the column separator. This will change the width to accommodate the widest item.

Each of the side lists has only one column. You can sort them the same way by clicking the column header, or move items using the ↑ and ↓ buttons at the top of the list. The connections will be laid out according to the order in the side lists — the top and bottom connections will be placed from left to right, and the left and right connections from top to bottom.

Move Label — (Also **<F6>**) This command activates the Move Label tool. This tool is used to relocate text labels. When the tool is active the mouse pointer becomes an .

Use this tool by clicking tacks on the line you want to move the label for. The label will jump to the current pointer location from which you can drag the label out to the new location.

Zoom In — (Also **<F7>**) This command activates the Zoom In tool. This tool increases the scale of the diagram by the scale increment set in the **Scale** dialog box (see Figure 5.11 on page 60). The default value of the scale increment is 20%. When this tool is activated the mouse pointer becomes a .

To zoom in click anywhere in the diagram. When the scale changes, the place on the diagram under the pointer is kept as constant as possible, providing the more de-

tailed view of the specified location.

Zoom Out — (Also **<F8>**) This command activates the Zoom Out tool. This tool decreases the scale of the diagram by the scale increment set in the **Scale** dialog box (see Figure 5.11 on page 60). The default value of the scale increment is 20%. When this tool is activated the pointer becomes a  .

To zoom out just click in the diagram. When the scale changes, the place on the diagram under the mouse pointer is kept as constant as possible, providing the wider view of the surroundings of the specified location.

Toggle Zoom — (Also **<F9>**) This command activates the Toggle Zoom tool. The tool switches the scale of the diagram between the current scale and the other one set in the **Scale** dialog box (Figure 5.11 on page 60) as the **Toggle** scale (100% by default). When this tool is activated and the current scale is smaller than the toggle scale the pointer becomes a  , if the current scale is larger than the toggle scale the pointer becomes an  .

To switch the scale click in the diagram. Click again to return to the previous scale. When the scale changes, the place on the diagram under the mouse pointer is kept as constant as possible, providing the more detailed or wider view of the specified location.

Navigate — (Also **<F10>**) This command activates the Navigate tool. This tool is used to quickly navigate page references and inheritance trees. When this tool is activated the mouse pointer becomes a 

When you click a page reference, the tool will either scroll the diagram immediately to make the referred box visible, or if the page reference box refers to multiple objects, the tool will display a menu or a list box containing references you can follow.

When you click an entity box, the tool will display the following menu:

- **Forward References —** Displays a menu or a list box containing all the forward page references of the selected entity.
- **Backward References —** Displays a menu or a list box containing all the backward page references of the selected entity.
- **Subtypes** — Displays a menu or a list box containing all the subtypes of the selected entity.

- **Supertypes** — Displays a menu or a list box containing all the supertypes of the selected entity.

If you click in the blank space, the tool will display the menu to navigate all the root objects in the diagram:

- **Inheritance Roots** — Displays a menu or a list box containing all the inheritance root objects (the objects that do not have supertypes).

- **Reference Roots** — Displays a menu or a list box containing all the reference root objects (the objects that are not attributes of other objects).

Reference and inheritance menus and lists are indented representations of a tree, where the greater indentation indicates a further distance in the selected direction. The punctuation in these lists corresponds to the punctuation used in STEP AP Mapping Tables (see table on page 49).

 **Page Reference —** (Also **<F11>**) This command activates the Page Reference tool that is used to insert page references. When the tool is activated the mouse pointer becomes a  .

To insert a page reference click the line you want to have a break in. It is also possible to break off all the children of an inheritance by clicking the tack where all the children split off (called a forked tack). Please note that there are several restrictions to where page references can be inserted. The two most common problems are: you can not insert a page reference before the forked tack, and you can not have more than one page reference on a line.

 **Remove —** (Also **<F12>**) This command activates the Remove tool. The Remove tool is used to remove unwanted tacks and page references from the diagram. When the tool is activated the mouse pointer becomes a  .

To use this tool simply click the object to be deleted. If the object is a page reference *onto* the page the tool will display a menu of references that may be deleted. Choose a reference from the menu and the reference count will be reduced by one. When the reference count reaches zero, the page reference box will be removed.

If the object is a page reference *from* the page the tool will display the following menu:

- **Delete Only** — Deletes the page reference box.

- **Delete and Yank** — Deletes the page reference box and moves (yanks) the referred box close to the referring box.

5.5.5 Color Menu

This menu has five color commands for each of the possible object colors. To change the color of an object, select it and then choose one of the menu colors.

It also has the **Working Set** command that is used to include the colored objects in the working set.

5.5.6 Window Menu

This menu contains commands for window control. EXPRESS-G Editor allows you to open several files at once, each of them placed in a separate window. The following commands are available through this menu:

Cascade — Arranges the windows in an overlapped fashion, so that all title bars are visible.

Tile — Arranges multiple opened windows vertically in a non-overlapped fashion.

Arrange Icons — Arranges the icons for minimized windows at the bottom of the main window. If there is an open document window at the bottom of the main window, then some or all of the icons may not be visible because they will be underneath this document window.

Split — Splits the active window into two panes. You may then use the mouse or the arrow keys to move the splitter bar. When the splitter bar is in the desired position, click or press **<Enter>** to set the splitter bar in its new position. Press **<Esc>** to keep the splitter bar in its original position.

Using split configuration you can view two parts of the same diagram at once, perhaps with different scales, so that the part with the smaller scale can serve as a map to the other part. Figure 5.14 shows a split window with two views of the same di-

agram.

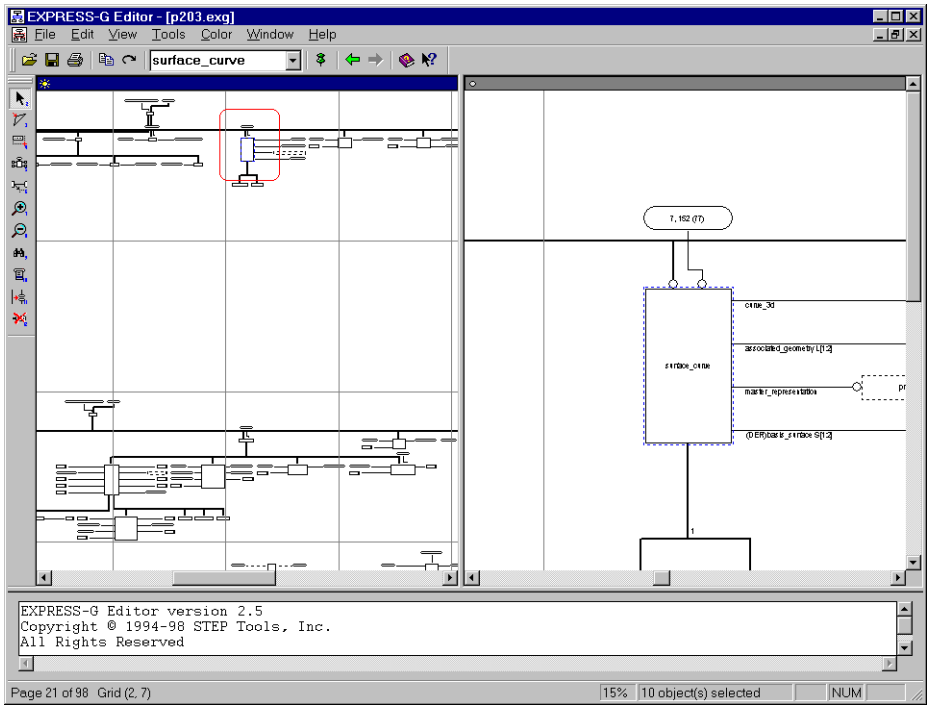


Figure 5.14 — Split Diagram Window

The area outlined in the rounded box on the left pane is shown on a large scale (150%) on the right pane.

Clear Messages — Clears the contents of the message window.

Window 1, 2... — EXPRESS-G Editor displays a list of currently open diagram windows at the bottom of the Window menu. A check mark appears next to the diagram name of the active window. Choose a window from this list to make it active.

5.5.7 Help Menu

Commands in this menu are used to access the help system.

EXPRESS-G Editor Help — (Also **<F1>**) This command displays the context-sensitive Help, which is the help information relevant to the current program context — active window, menu, dialog box, etc.

Contents and Index — (Also **<Ctrl+F1>**) Displays the Help browser (see Figure 5.15). The Help browser is used to navigate through Help topics using table of contents, index, text search, and navigation buttons.

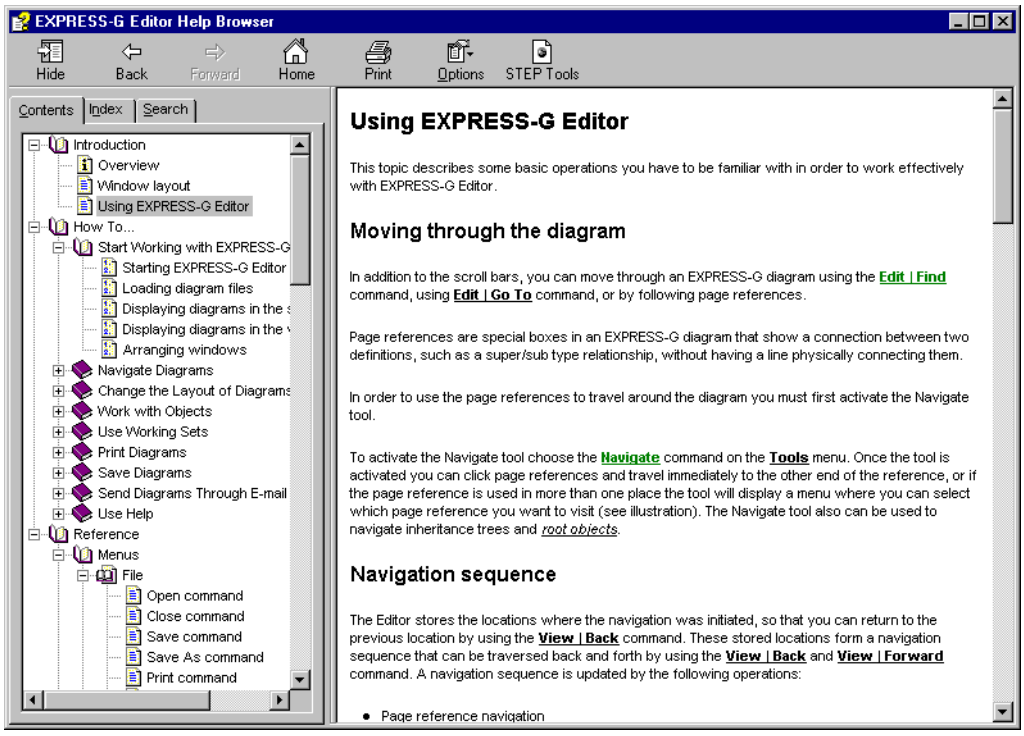


Figure 5.15 — Help Browser

What's This? — (Also **<Shift+F1>**) This command puts EXPRESS-G Editor into the Help mode.

This command will change the mouse pointer to an ? to indicate that the editor is in the Help mode. Click a menu command, toolbar button, or click in a part of the main window to get information about this command, button, or window part.

STEP Tools on the Web — Displays a menu with the following commands:

- **STEP Tools Home Page** — Starts your default browser and connects to the STEP Tools, Inc. homepage.
- **Products** — Starts your default browser and connects to the STEP Tools, Inc. products page.
- **Support** — Starts your default browser and connects to the STEP Tools, Inc. technical support page.

About EXPRESS-G Editor — Displays the copyright notice, version number and the license information of your copy of EXPRESS-G Editor.

5.5.8 Shortuct Menus

Shortcut menus appear when you right-click or press the Application key ( ). There are two types of shortcut menus provided by EXPRESS-G Editor — the Object menu and the Diagram menu.

The Object Shortcut Menu

The Object menu appears when you right-click an object in the diagram. This menu has the following commands:

Copy — Copies the selected object(s) to the Clipboard.

Edit Bookmark — Creates a text input box to edit the selected bookmark (This command is on the menu only if the selected object is a bookmark.)

Delete Bookmark — Deletes the selected bookmark. (This command is on the menu only if the selected object is a bookmark.)

Add to Working Set — Adds selected entities/types to the working set. (This command is on the menu only if the selection contains entity and/or type boxes.)

View Working Set — Displays the **Working Set** dialog box (see Figure 5.12 on page 60) to view, edit, and apply the working set. (This command is on the menu only if the selection contains entity and/or type boxes.)

Select — Activates the Select tool.

Add Tack — Activates the Add Tack tool.

Resize — Activates the Resize tool.

Layout — Activates the Layout tool.

Move Label — Activates the Move Label tool.

Zoom In — Activates the Zoom In tool.

Zoom Out — Activates the Zoom Out tool

Toggle Zoom — Activates the Toggle Zoom tool.

Navigate — Activates the Navigate tool.

Page Reference — Activates the Page Reference tool.

Remove — Activates the Remove tool.

Color — Opens the **Color** menu (see Section 5.5.5 on page 66).

Scale — Displays the **Scale** dialog box (see Figure 5.11 on page 60).

All Diagram — Scales the diagram to fit everything into the diagram window.

Full Size — Displays the diagram in full size (scale of 100%).

EXPRESS Source — Displays the **EXPRESS Source** dialog box with the EXPRESS language definition of the currently selected object. (This command is on the menu only if the selected object is an **ENTITY** or **TYPE** box.

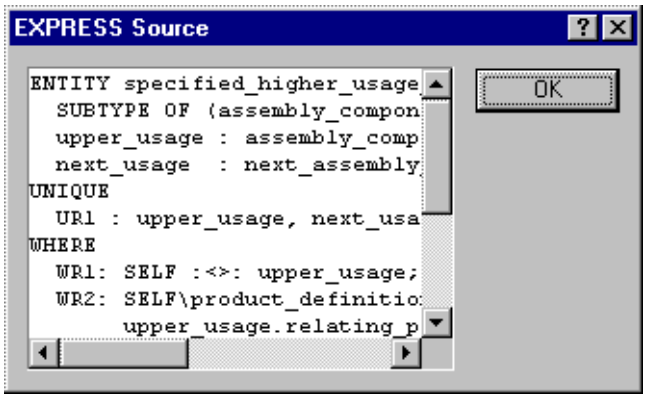


Figure 5.16 — EXPRESS Source Dialog Box

Show this on the other pane — Scroll the other pane to show the same place on the diagram that is being shown on the active pane. (This command is on the menu only if the active diagram window is split.)

This command provides a quick way to see a specific place in a diagram. When the window is split and one pane has larger scale than the other, and you click on a diagram in the pane with smaller scale, then if you select this command the same place of the diagram will be shown on the other pane with the larger scale providing more detailed view.

The Diagram Shortcut Menu

The Diagram menu appears when you right-click in the blank space of the diagram. This menu has the following commands:

Show this on the other pane — See above.

Select — Activates the Select tool.

Add Tack — Activates the Add Tack tool.

Resize — Activates the Resize tool.

Layout — Activates the Layout tool.

Move Label — Activates the Move Label tool.

Zoom In — Activates the Zoom In tool.

Zoom Out — Activates the Zoom Out tool

Toggle Zoom — Activates the Toggle Zoom tool.

Navigate — Activates the Navigate tool.

Page Reference — Activates the Page Reference tool.

Remove — Activates the Remove tool.

Scale — Displays the **Scale** dialog box (see Figure 5.11 on page 60).

All Diagram — Scales the diagram to fit everything into the diagram window.

Full Size — Displays the diagram in full size (scale of 100%).

Properties — Displays the **Diagram Properties** dialog box (see Figure 5.9 on page 57).

5.5.9 Quick Scale Menu

This menu is displayed when you click the scale indicator  located on the status bar.

The Quick Scale menu contains the following commands.

200% — Changes scale to 200% (2 times magnification).

150% — Changes scale to 150% (1 1/2 times magnification).

125% — Changes scale to 125% (1 1/4 times magnification).

100% — Changes scale to 100% (normal size).

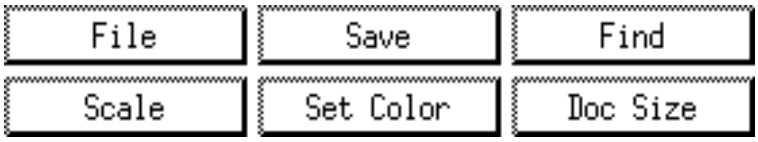
75% — Changes scale to 75% (3/4 of normal size).

50% — Changes scale to 50% (1/2 of normal size).

25% — Changes scale to 25% (1/4 of normal size).

Fit — Changes the scale to fit the whole diagram into the window.

5.6 Menus (UNIX Version)



The menu commands do useful things to the document, such as changing the number of pages, the current scale, or finding an entity or attribute. Menu options that have ellipses '...' after them will bring up a dialog box.

From this menu you can open, close, and save EXPRESS-G diagrams or you can exit the application.

Save — Save the EXPRESS-G document in the window that the menu item was selected from. This overwrites the old version of the document.

Close — Close the EXPRESS-G window.

Open... — Open an EXPRESS-G file in a new window.

Exit... — Close all EXPRESS-G windows and exit.

Save any changes to the document. Overwrites the old version of the document. This is equivalent of the **Save** comand under the **File** menu.

This will search for objects using the criteria in the find edit box. Hitting the return key in the edit box will produce the same results. The object that matches the string will be surrounded by red. In addition the editor will center the object in the middle of the scrolled area.

Scale

Change the viewing magnification. The last item in the menu allows you to adjust the resolution exactly.

2.0 (double size) ... 0.125 (1/8th size) — View the diagram at a range of predefined magnifications.

Fit all on Screen — Adjust the resolution so the entire diagram is visible.

Any Scale... — View diagram at an arbitrary magnification. This operation also provides direct access to the alternate scale used by the zoom tool.

Set Color

Apply colors to objects in the document. When a color is chosen, all currently selected items are given the color.

No Color - Sets selected items to black (foreground color)

Color 1 (Blue) - Sets selected items to blue.

Color 2 (Green) - Sets selected items to green.

Color 3 (Red) - Sets selected items to red.

Color 4 (Gray) - Sets selected items to gray.

Doc Size

This brings up a menu where you can add and delete pages from the right and bottom of the document. Note that the entire row or column of pages must be empty in order to delete it.

Add Row - This will add a row of pages along the bottom of the document.

Add Column - This will add a column of pages along the right edge of the document. You should take note of the fact that this will change the page numbers on all of the pages (and page references) for all pages after the first row.

Delete Row - This will delete a row from the bottom of the document. For this operation to succeed you must move all objects out of the row to be deleted.

Delete Column - This will delete a column from the right edge of the document. For this operation to succeed you must move all objects out of the row to be deleted. Also be aware that this will change most of the page numbers.

5.7 Toolbars

5.7.1 Windows Toolbar

The Windows version has three toolbars: the standard toolbar normally placed across the top of the main window, under the menu bar, the **Diagram Tools** toolbar normally placed at the left side of the main window, and the Print Preview toolbar, which is visible only in the print preview mode (see Figure 5.5 on page 54). The standard and **Diagram Tools** toolbars can be attached to any of the four sides of the window, or left floating free on the desktop.

To move a toolbar to a new location, click the toolbar handle  , and then drag the toolbar. While dragging, you will see the outline of the toolbar. If this outline is thick, it means that releasing the mouse button will leave the toolbar floating, if this border is thin, the toolbar will be attached to the nearest window side.

The toolbar buttons provide quick mouse access to menu commands. The toolbar buttons and their corresponding commands are described below.

Standard Toolbar

The standard toolbar contains basic commands from the **File**, **Edit** and **Help** menus.

Click	To
	Open an existing diagram.
	Save the active diagram.
	Print the active diagram.
	Copy selected objects to the Clipboard.
	Find the next box matching the pattern.
	Type the search pattern.
	Insert bookmarks in the diagram.
	Go to the previous location in the navigation sequence.
	Go to the next location in the navigation sequence.

Click	To
	Display the Help browser.
	Put the editor into the Help mode.

The Diagram Tools Toolbar

This toolbar provides quick mouse access to the diagram tools in EXPRESS-G Editor.

Click	To
	Activate the Select tool.
	Activate the Add Tack tool.
	Activate the Resize tool.
	Activate the Layout tool.
	Activate the Move Label tool.
	Activate the Zoom In tool.
	Activate the Zoom Out tool.
	Activate the Toggle Zoom tool.
	Activate the Navigate tool.
	Activate the Page Reference tool.
	Activate the Remove tool.

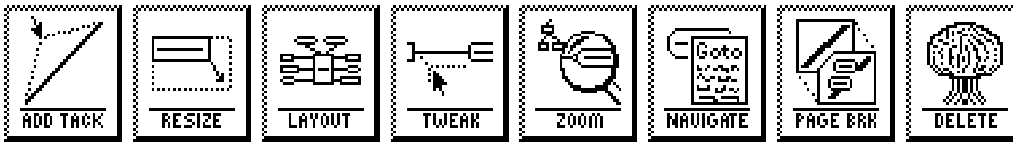
The Print Preview Toolbar

The Print Preview toolbar offers the following commands.

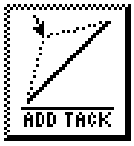
Click	To
Print	Display the **Print** dialog box and start a print job.
Next Page	Preview the next printed page.
Prev Page	Preview the previous printed page.

Click	To
One Page/ Two Page	Preview one or two printed pages at a time.
Zoom In	Take a closer look at the printed page.
Zoom Out	Take a larger look at the printed page.
Close	Return from print preview to the editing window.

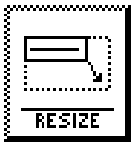
5.7.2 UNIX Toolbar



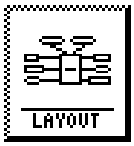
The tool bar is located at the upper left of the window. When the program first starts up the **Zoom** tool will be selected. You can always tell what tool is selected because it will appear inverted from the rest of the tool buttons. The selected tool takes over the second (middle) mouse button.



The **Add Tack** tool creates new tacks on existing lines. Press the mouse button with the pointer over the line which you wish to add a tack to. Then while still holding the mouse button down drag the pointer to the desired location for the new tack. Then release the mouse button.

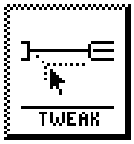


The **Resize** tool resizes boxes. Press the mouse button inside of a box and then drag the outline to the desired size.

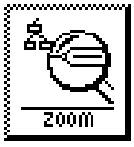


The **Layout** tool rearranges the lines and boxes so they looks similar to the layout originally created by **express2expg**. Press the mouse button over an entity. A menu will appear with choices for placing all attributes on the left, all on the right or on alternate sides. Select an option and release the mouse button.

This tool can also clean up lines. Press the mouse button on a line. A menu will appear that offers you the choice of laying out the entire line or just the segment you clicked on. If you select the segment, it will update the line segment between the two closest points. If you select the whole line, it will delete any tacks you may have added and then re-layout the line. If you press the mouse button on a forked tack it will layout all of the surrounding lines.



The **Tweak** tool repositions a label for a line. This is done by pressing the mouse button when the mouse is over any of the tacks on the line you want to move the label for. The label will jump to the current pointer location from which you can drag the label out to the desired new location.



The **Zoom** tool toggles between two resolutions (the resolutions can be set by using the **Any Scale** menu option under the **Scale** menu talked about above, they default to 1.0 and fit all on screen). Also it will highlight the box, line or point that was under the mouse when the button was pressed.

The **Page Reference Navigation** tool traverses page references. When you click on a page reference you either immediately move to the box the page reference refers to, or if the page reference box refers to multiple objects a menu is brought up which allows you to select which reference you want to follow.

The **Page Break** tool inserts page references. Click on the line you want to break. It is also possible to break off all the children of an inheritance by simply clicking on the tack where all the children split off (called a forked tack). There are some restrictions to where page breaks can be inserted. In particular, you can not insert a page break before the forked tack, and you can not have more than one page break on a line.

The **Delete** tool allows you to remove unwanted tacks and page breaks from the diagram. Simply press the mouse button when the mouse is over the entity to be deleted. In the case of a page reference box if there are multiple objects that reference it you will be given a menu of objects that may be deleted.

When the page reference refers to another entity a menu will give you the option of simply deleting the page reference box, or deleting the box and moving (or yanking) the object it refers to, to the location of the page reference (be careful when deleting and yanking).

5.8 Keyboard Shortcuts

The following keyboard commands can be used with the Windows version of EXPRESS-G Editor.

Use	To
Left Arrow	Scroll left one position (the same as the ◂ scrollbar button). If **<Scroll Lock>** is on, move the mouse pointer left one pixel instead of scrolling.
Alt+Left Arrow	Go to the previous location in the navigation sequence.

Use	To
Right Arrow	Scroll right one position (the same as the scrollbar button). If **<Scroll Lock>** is on, move the mouse pointer right one pixel instead of scrolling.
Ctrl+Right Arrow	Find the next box matching the search pattern.
Alt+Right Arrow	Go to the next location in the navigation sequence.
Up Arrow	Scroll up one position (the same as the scrollbar button). If **<Scroll Lock>** is on, move the mouse pointer up one pixel instead of scrolling.
Down Arrow	Scroll down one position (the same as the scrollbar button). If **<Scroll Lock>** is on, move the mouse pointer down one pixel instead of scrolling.
Home	Scroll left one screen. If **<Scroll Lock>** is on, move the mouse pointer left ten pixels instead of scrolling.
Ctrl+Home	Scroll all the way to the left.
End	Scroll right one screen. If **<Scroll Lock>** is on, move the mouse pointer right ten pixels instead of scrolling.
Ctrl+End	Scroll all the way to the right.
Page Up	Scroll up one screen. If **<Scroll Lock>** is on, move the mouse pointer up ten pixels instead of scrolling.
Ctrl+Page Up	Scroll all the way to the top.
Page Down	Scroll down one screen. If **<Scroll Lock>** is on, move the mouse pointer down ten pixels instead of scrolling.
Ctrl+Page Down	Scroll all the way to the bottom.
Ctrl+Insert	Copy the selection to the Clipboard.
Spacebar	Simulate the mouse click if the **<Scroll Lock>** is on.
Ctrl+Tab	Switch to the next open diagram window.
Ctrl+Shift+Tab	Switch to the previous open diagram window.
Ctrl+A	Select all objects in the diagram.
Ctrl+C	Copy the selection to the Clipboard.
Ctrl+F	Find a box using a search pattern.
Ctrl+G	Go to a page or bookmark.
Ctrl+L	Scale the diagram to fit it whole into the window.

Use	To
Ctrl+O	Open an existing EXPRESS-G diagram.
Ctrl+P	Print the active diagram.
Ctrl+S	Save the active diagram to the same file it was opened from.
Ctrl+R	Show or hide the **Panner** window.
Ctrl+U	Set the diagram scale to 100%.
Ctrl+W	View and edit the working set.
F1	Get help on the current context.
Ctrl+F1	Display the Help browser.
Shift+F1	Put the editor into the Help mode.
F2	Activate the Select tool.
F3	Activate the Add Tack tool.
F4	Activate the Resize tool.
Ctrl+F4	Close the active diagram window.
Alt+F4	Quit the editor.
F5	Activate the Layout tool.
F6	Activate the Move Label tool.
Ctrl+F6	Switch to the next open diagram window.
Ctrl+Shift+F6	Switch to the previous open diagram window.
F7	Activate the Zoom In tool.
F8	Activate the Zoom Out tool.
F9	Activate the Toggle Zoom tool.
F10	Activate the Navigate tool.
Shift+F10	Display the shortcut menu.
F11	Activate the Page Reference tool.
F12	Activate the Remove tool.
Alt+Enter	Edit diagram properties.
Application key ()	Display the shortcut menu.

5.9 Customizing

This section applies only to the UNIX version of the EXPRESS-G editor.

The **expgedit** editor is an Xt application. Any Xt application can be visually customized through the use of resource files. A sample resource file can be found in **$ROSE/etc/Expgedit**. This file describes the defaults that are compiled into the tool. You can customize this file and then tell the X server to use it by typing:

```
% xrdb -merge <filename>
```

Or set the **$XAPPLRESDIR** environment variable to directories containing resource files:

```
% setenv XAPPLRESDIR /directory_one:/directory_two
```

If you have some prior experience with Xt, you may find the widget tree for the tool to be useful:

```
Expgedit (applicationShell)
       TopLevelShell
              main (form)
              titleBar (box)
                     tools (box)
                            createTack_button (command)
                            boxResize_button (command)
                            layoutBox_button (command)
                            tweakLabel_button (command)
                            zoomView_button (command)
                            pageRefNavigation_button (command)
                            createPageRef_button (command)
                            deleteObj_button (command)
                     buttons (box)
                            file (menu button)
                            save (command)
                            find (command)
                            scale (menu button)
                            color (menu button)
                            docsize (menu button)
              editLabel (label)
              edit (text)
              scrolled (viewport)
                     canvas (simple)
                            menu (simpleMenu)
                                   entry0 (smeBSB)
                                   entry1 (smeBSB)
                                      .
              infoLabel (label)
```

6 EXPRESS-G Layout Engine

6.1 Description

The EXPRESS-G Layout tool will take syntactically correct EXPRESS text files and convert them to EXPRESS-G diagrams. The diagrams can then be viewed with **expgedit** or printed using either **expg2ps** or **expg2hpgl**.

6.2 Command Line

express2expg [options] expfile1 [expfile2 ...]

The options recognized by the diagram layout tool are shown below:

-help Print this list of options. The tool performs no other action and ignores all other options.

-cuttopage Force the layout engine to break definitions into pieces small enough to fit on a single page (where possible). This is useful for printed diagrams, but for interactive viewing it is more useful to preserve the full inheritance tree structure without the extra page breaks this creates.

-diagonals Allow the layout engine to use diagonal lines. By default, the engine uses a Manhattan layout (all lines at right angles).

-direct Direct inter-schema links. Normally, **express2expg** will attempt to keep schema's independent by inserting inter-schema references. This option will force the tool to link objects directly where possible. Because the diagrams may be spread out, this tends to just replace the inter-schema references with page breaks. Use the **-nobreak** or **-farbreak** options to disable page breaks.

-farbreak Don't insert a page break if objects are "close." When arranging a diagram, the tool inserts page breaks to avoid long crossing lines. This option forces direct connections instead of page breaks, for objects that are near one another. This is useful for smaller schemas, where it is possible to work with most of the schema at once. Note, however, that a large number of lines can slow down display with the **expgedit** editor.

-fontname <name>
Set the font for printing. The default is Helvetica-Bold. In order for **express2expg** to properly layout the boxes for printing it needs to have a description of the font. On UNIX systems we use font metrics files that are freely available from Adobe. See **UNIX Technical Notes** (Section 6.6, pp. 87) for more information about obtaining, and adding font metrics files to your installation.

-fontsize <point-size>
Set font size for printing. The default is 7 pt. text. This can be too small for some purposes, but is quite readable, and reproducible on a 300dpi printer. Due to limitations in X11, the full scale view of your document will always use a 10 pt. font. Page breaks and boxes will be scaled according to this value.

-fulltree Keep inheritance trees full. The **express2expg** tool normally breaks up wide inheritance trees by inserting page breaks. You may want to use this option when preparing EXPRESS-G diagrams for printing with either **expg2ps** or **expg2hpgl**.

-i / -index
Print an index listing for every definition in the diagram. The name of each definition is printed, along with the number of the page it is defined on. The **expginfo** tool can also produce such a listing. See **EXPRESS-G Diagram Information** (Chapter 9, pp. 97) for an example.

-l / -landscape
Swap the x and y dimensions of the page. For example, if you had 8.5x14 (legal-sized printer) paper, and wanted landscape mode you would use: **express2expg -pagewidth 7.5 -pageheight 13 -l**

-nobreak Do not insert page breaks in the diagram. This is only useful on rather small schemas.

-pagesize <std-size>
Select one of the standard ANSI paper sizes **A**, **B**, **C**, **D**, **E**, or or ISO paper sizes **A0**-**A4**

-pageheight <height-in-inches>
Set the size in inches of the height of the page. The default is 10.0". See **-pagewidth** above for some warnings as to its use.

-pagewidth <width-in-inches>
Set the size in inches of the width of the page. The default is 7.5". If you have a printer that takes larger paper you may wish to change this. Note that you often need to leave a margin around the page where the printer is incapable of printing.

-ref <num> Change the starting number of the page references. Page references will start at **<num>** and increase by 1. Note that page references are not the same as page numbers. Each page reference has a unique number within the EXPRESS-G diagram. This offers a way to have unique page references through multiple EXPRESS-G diagrams.

-v Verbose mode will give more information about the conversion process, such as messages from the EXPRESS compiler. It is also useful for tracking the progress through large schemas.

-workingset <namefile>
Generate EXPRESS-G for the subset of entities in a working set. Reads a list of names from **<namefile>** and creates diagrams for just those entities and supporting types. Useful when working with a few types or entities from a large APs. See **Customizing Output with a Working Set File** (Section 2.9, pp. 24) for a description of working sets.

6.3 Windows Control Panel

The EXPRESS-G Layout Windows control panel is shown in Figure 6.1. Run this by selecting **EXPRESS-G to EXPRESS-G** in the ST-Developer Launcher or the "ST-Developer Tools" Start Menu folder. The following sections describe fields and setting on this control panel and show how to control the converter.

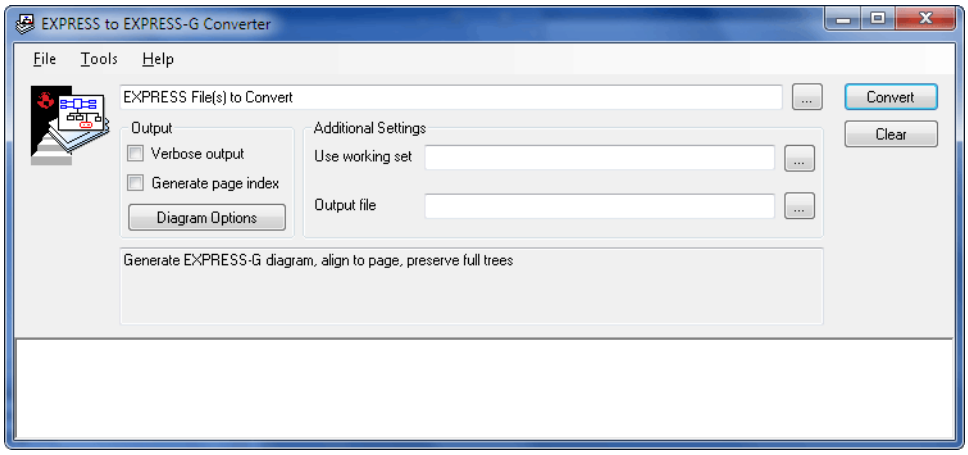


Figure 6.1 — EXPRESS-G Layout Windows Control Panel

6.4 Generating Diagrams

At the top of the EXPRESS-G Layout control panel shown in Figure 6.1 is a text field for one or more schema files that you would like to convert. You can open the file dialog using **Ctrl+O** or the **[...]** button to the right. You can also drag and drop files from the Windows Explorer.

The **Use working set** field or **-ws** command line option can specify a file containing a subset of definitions that should appear in the diagram. For more information on working sets see the *ROSE Library Reference Manual*.

The **Output file** field or **-f** command line option provide the name of the resulting EXPRESS-G diagram file. If do not specify the name of the output file, the diagram file will be the same as the first EXPRESS file, but with the **'.exg'** extension.

The **Verbose output** option or **-v** flag make the tool display more information about the conversion process and messages from the EXPRESS compiler.

The **Generate page index** option or **-index** flag prints an index which lists the location of all definitions.

The **Diagram Options** button brings up the dialog shown in Figure 6.2 with additional options for controlling the layout process. In the **Layout** area the **Break diagram lines** menu controls how long lines are separated by page references. The **Only between far objects** choice and **-farbreak** flag insert fewer page reference boxes, while the Never choice and -nobreak flag insert no page references.

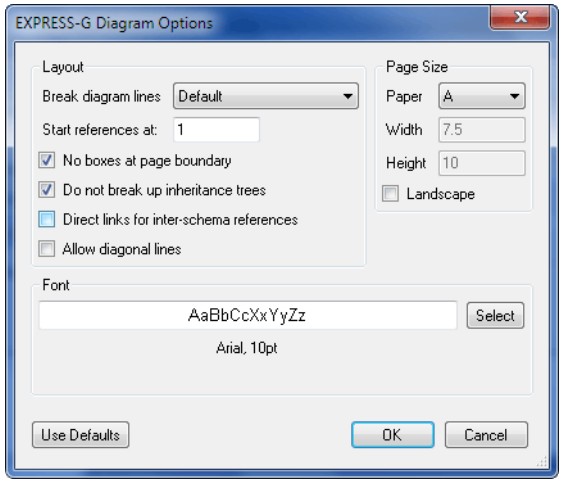


Figure 6.2 — EXPRESS-G Layout Diagram Options

The **Start references at** field and **-ref** command line option give a numeric starting value for the references boxes in the diagram. References start at this number and increase by 1. Each page reference has a unique number within an EXPRESS-G diagram. This offers a way to keep unique references through multiple diagrams.

The **No boxes at page boundary** option or the **-cuttopage** flag breaks up definitions until they are small enough to fit on a single page (where possible).

The **Do not break up inheritance trees** option or the **-fulltree** flag keeps inheritance trees full. Use this option to create wall-sized EXPRESS-G diagrams for printing.

The **Direct links for inter-schema references** option or the **-direct** flag forces the tool to link objects directly where possible. Normally, the layout engine attempts to keep schemas independent by inserting inter-schema references.

The **Allow diagonal lines** option or the **-diagonal** flag removes the restriction on the connecting lines to horizontal and vertical (Manhattan style) layout.

The **Page Size** area describes the paper to assume when placing and grouping the diagram elements. The **Paper** field or **-pagesize** option selects from a list of standard ANSI and ISO paper sizes. The **Width** / **Height** fields or **-pagewidth** / **-pageheight** flags are used for custom page sizes. The **Landscape** option or **-landscape** flag swaps the height and width page dimensions.

The **Font** field or **-fontname** / **-fontsize** flags specify the font to be used by the diagram. The options dialog shows a sample of the font. Click the **Select** button to pick a font using the Windows font dialog.

6.5 Layout Style

The layout engine considers the inheritance hierarchy first, and will arrange definitions so that references to an entity come in from the top and references to sub-types are placed at the bottom of an entity. Attributes will be placed at the sides of each entity.

On the virtual page, the largest inheritance trees are placed first. The converter proceeds from left to right, top to bottom across the virtual page. The largest tree is in the upper left hand corner and the smallest is at the bottom, as shown in Figure 6.3. The smallest objects tend to be defined types, selects, enumerations or unused entities.

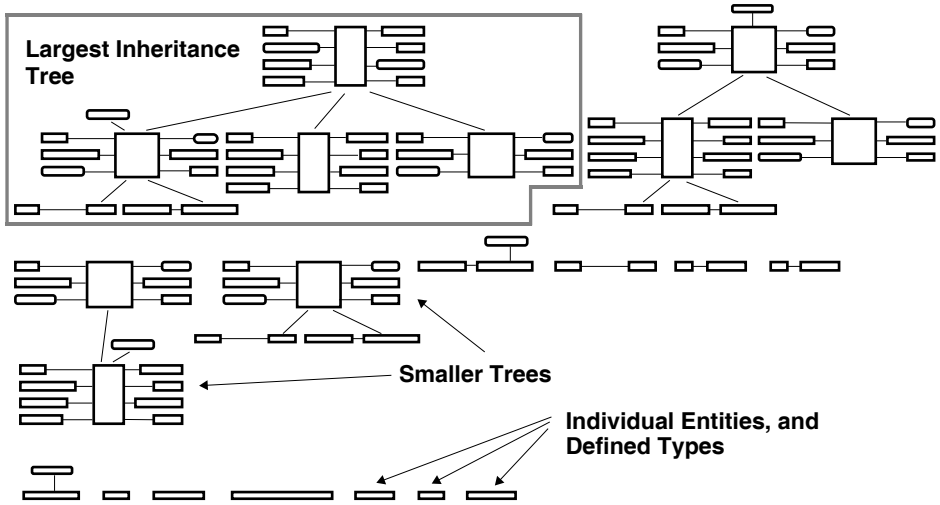


Figure 6.3 — Sample Diagram Layout

The default layout style is useful for learning or comparing EXPRESS schemas. By default the tool will try to keep the inheritance trees whole. The tool inserts a page reference or an inter-schema reference when ever it is unable to put the actual entity in that location. The tool disregards page boundaries when placing entities.

You may wish to change the defaults when preparing diagrams for printed documentation. For instance, use the **-cuttopage** option to break the diagram into page-sized pieces.

6.6 UNIX Technical Notes

The UNIX EXPRESS-G tools use Adobe font metrics files to calculate basic information about the font sizes. Without a proper font metrics file **express2expg** will tend to generate a poor layout. Adobe provides font metrics files via anonymous FTP (**ftp://ftp.adobe.com/pub/adobe/type/**). We have included files for Helvetica, Times, and Courier in normal, italic, bold and bold&italic styles. The files are installed in **$ROSE/lib/afm**.

The EXPRESS-G tools will check for the desired font metrics file in the following places:

```
$ROSE_AFM  (a list of colon separated directories to check)
$HOME
$HOME/afm
$ROSE/system_db/afm
```

7 EXPRESS-G Diagram Update

7.1 Description

The **expgupdate** tool updates an EXPRESS-G diagram with a new EXPRESS schema. This tool preserves your EXPRESS-G layout when you make small changes to your original EXPRESS text. It compares the diagram and new schema and will produce a new diagram with the original layout information and new EXPRESS definitions.

7.2 Command Line

expgupdate <expg-diagram> <express-schema>

The diagram update tool accepts the following options:

-help Print this list of options. The tool performs no other action and ignores all other options.

-v / -verbose
Give more information about the conversion process. This will print all of the matches that it found, (and the likelihood that the two are supposed to be matched).

7.3 Windows Control Panel

The EXPRESS-G Update Windows control panel is shown in Figure 7.1. Run this by selecting **EXPRESS-G Update** in the ST-Developer Launcher or the "ST-Developer Tools" Start Menu folder. The following sections describe the fields and setting on this control panel.

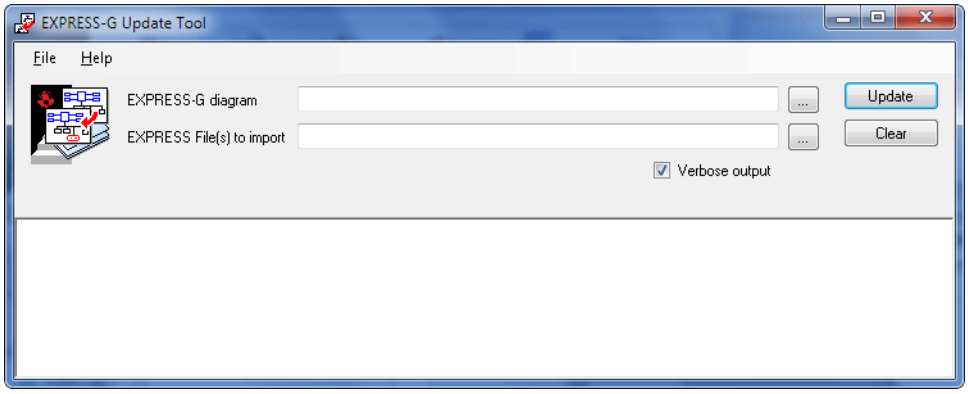


Figure 7.1 — EXPRESS-G Update Control Panel

7.4 Updating an EXPRESS-G Diagram

At the top of the EXPRESS-G Update control panel is a text field for the EXPRESS-G diagram. Below that is a text field for one or more EXPRESS files that you would like to compile. You can open the file dialog using **Ctrl+O** or the **[...]** button to the right of either field. You can also drag and drop files from the Windows Explorer and they will go into the correct field based on their file extension.

The **Verbose** output option or **-v** flag will display information about matches that it found, and the likelihood that the two are supposed to be matched.

When you run the tool, it will change the EXPRESS-G to reflect changes you have made in the EXPRESS file. The **expgupdate** tool is useful for preserving your layout work. It is common to spend a fair amount of time arranging a diagram, for this reason, it is important to be able to preserve your layout when small changes are made in the underlying EXPRESS.

The example below uses the **expgupdate** tool to incorporate new schema definitions and carry over graphics from an existing EXPRESS-G diagram. In this case we have an existing diagram **AP210.exg** and original EXPRESS source **AP210.exp**.

```
% expgupdate AP210.exg  AP210.exp
```

The tool first generates the EXPRESS-G for the new AP210. This EXPRESS-G file will have the default layout. Next, the tool will compare the two schemas and repeat any changes you made to **AP210.exg** wherever possible. The result of this operation may still need some hand-tuning, but most of the layout will be carried over.

8 EXPRESS-G Diagram Compare

8.1 Description

The **expgdiff** tool compares two EXPRESS-G diagrams. It can report results in two different ways. It can apply color to the entities that the two schemas share or color all of the entities that differ between the two schemas.

8.2 Command Line

expgdiff [options] <old-expg> <new-expg>

The EXPRESS-G diagram comparison tool accepts the following options:

-help Print this list of options. The tool performs no other action and ignores all other options.

-cs / -colorsim (default)
Apply **color1** (blue) to all entities that appear in both old and new EXPRESS-G diagrams. This will overwrite the original files, so you might want to be careful if you have colored some of the entities yourself.

-cd / -colordiff
Highlight all of the entities that are different between the old and new EXPRESS-G diagrams. Deleted entities will be colored red

(color3), new entities will be colored green (color2). This will overwrite the original files, so you might want to be careful if you have colored some of the entities yourself.

-retainold
Prevent the tool from changing the old diagram.

-setaddcolor <color>
Specify the color for objects added to the new diagram.

-setdeletecolor <color>
Specify the color for objects deleted from the old diagram.

-setmatchcolor <color>
Specify the color for objects shared between the two diagrams.

-v / verbose
Give more information about the conversion process. This will print all of the matches that it found, (and the likely hood that the two are supposed to be matched).

The color specifications used for options can be any of the following. The numbers are important since it is possible to override the color values in an **Xdefaults** file for the UNIX version of **expgedit**, and they also correspond to the pen number used in HP-GL devices.

```
0, black, blk, none, normal
1, green, grn
2, blue, blu
3, red
4, gray, gry
```

8.3 Windows Control Panel

The EXPRESS-G Difference Windows control panel is shown in Figure 8.1. Run this by selecting **EXPRESS-G Difference** in the ST-Developer Launcher or the "ST-Developer Tools" Start Menu folder. The following sections describe the fields and setting on this control panel.

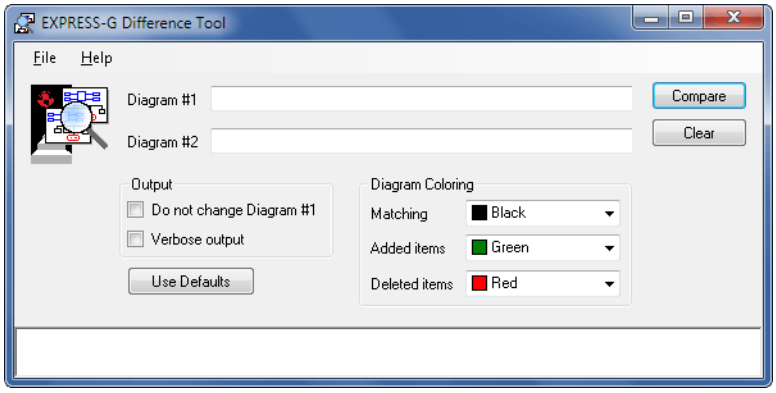


Figure 8.1 — EXPRESS-G Difference Control Panel

8.4 Comparing Diagrams

At the top of the EXPRESS-G Difference control panel are text fields for two EXPRESS-G diagrams. You can open the file dialog using **Ctrl+O** or the **[...]** button to the right of either field. You can also drag and drop files from the Windows Explorer into the text fields.

When you run the tool, these EXPRESS-G files will be colored to indicate similarities and/or differences between them. Any old color information in an EXPRESS-G file is lost when the tool is run.

The **Do not change Diagram #1** option or the **-retainold** flag prevents any changes to the first diagram. Only the second diagram will be updated with new colors.

The **Verbose** output option or the **-v** flag gives more information about the comparison process. This will display all matches found and the likelihood that the two are supposed to be matched.

The **Diagram Coloring** area controls the colors for matching, added and deleted objects. Each of the lists contains 5 colors — black, blue, green, red, and gray. From the command line, these can be controlled by the **-setaddcolor**, **-setmatchcolor**, and **-setdeletecolor** flags.

The **expgdiff** tool simplifies the comparison of large EXPRESS schemas. The tool considers two entities, attributes, selects, or enumerations to be the same if they have the same or similar name. When it compares names, it looks for common substrings. For example, it would find a strong match between “location” and “new-

location." This tool only examines the names of attributes, and therefore will not detect a change in the underlying type of an attribute.

The **expgdiff** tool uses the same comparison engine as **expgupdate**. Since **expgupdate** was the driving force behind this utility, the differencing capabilities are somewhat limited (because **expgupdate** is oriented towards finding similarities).

The following example shows how one might use the **expgdiff** tool to find the similarities between STEP Part 42 and Part 203.

```
% express2expg part-42.exp
% express2expg part-203.exp
% expgupdate part-42.exg part-203.exg -v
```

The first two lines generates the EXPRESS-G for the schemas. The third line compares the two schemas and highlights all of the entities the two models share.

9 EXPRESS-G Diagram Information

9.1 Description

The EXPRESS-G information tool **expginfo** is a command line utility that can extract information from an EXPRESS-G diagram.

9.2 Command Line

expginfo [options] <expg-file>

The EXPRESS-G Information tool accepts the following options. Without any options, the tool produces both size and index listings.

-help Print this list of options. The tool performs no other action and ignores all other options.

-i / -index Print an index listing for every definition in the EXPRESS-G diagram. The name of each definition is printed, along with the number of the page it is defined on.

-s / -size Print page dimensions, page count, and grid arrangement of the EXPRESS-G diagram.

9.3 Overview

Without any options, the tool produces both size and index listings. The following example shows the contents of an AP-203 EXPRESS-G diagram:

```
% expginfo part-203.exg

Pages are 7.5" wide by 10.0" high.
There are 112 pages, arranged in a grid which is
        16 pages wide by 7 pages high.

action,65
action_assignment,76
action_execution,65
action_method,84
action_request_assignment,73
action_request_status,104
action_status,105
address,58
advanced_brep_representation,61
ahead_or_behind,87
alternate_product_relationship,94
application_context,71
application_context_element,72
[ . . . ]
```

10 EXPRESS-G to PostScript Converter

10.1 Description

The **expg2ps** tool converts EXPRESS-G diagrams to PostScript page descriptions. The output may be sent to a file or piped directly to a printer or another program.

This tool is intended primarily for the UNIX platform, but is also available on Windows. Windows users can also print directly from the EXPRESS-G editor.

10.2 Command Line

expg2ps [options] <expg-file>

The EXPRESS-G to PostScript converter accepts the following options:

-c / -color
: Generate a color PostScript file. This is optional as some older printers may be unable to process color PostScript commands.

-f / -file <output-file>
: Write PostScript data to a file. By default the tool writes to standard output.

-pages <start>[-<end>]
: Select one or more pages to print. Page numbers can be found using

the EXPRESS-G editor. By default the tool converts the entire diagram.

-v Give more information about the conversion process. It is useful for tracking the progress through large schemas.

10.3 Overview

The **expg2ps** tool converts an EXPRESS-G file to a stream of PostScript, which is normally piped to standard output. The generated PostScript layout is identical to the layout that can be seen in **expgedit**, except that it is broken into multiple pages. One other important difference is that boxes will always appear on top of lines. This is not true in **expgedit** for performance reasons.

By default, the PostScript is piped to standard output. This can be overridden by simply using the **-file** option described above.

```
% expg2ps diagram.exg > outfile.ps
% expg2ps -file outfile diagram.exg

% expg2ps diagram.exg | lpr
```

11 EXPRESS-G to HP-GL Converter

11.1 Description

The **expg2hpgl** tool converts EXPRESS-G diagrams to HP-GL plotter commands. The output may be sent to a file or piped directly to a plotter or another program.

This tool is intended primarily for the UNIX platform, but is also available on Windows. Windows users can also print directly from the EXPRESS-G editor.

11.2 Command Line

expg2hpgl [options] <expg-file>

The EXPRESS-G to HP-GL converter accepts the following options.

-color1pen <pen-number>
This allows the user to set the 'color1' or blue pen (default 2).

-color2pen <pen-number>
This allows the user to set the 'color2' or green pen (default 3).

-color3pen <pen-number>
This allows the user to set the 'color3' or red pen (default 4).

-color4pen <pen-number>
This allows the user to set the 'color4' or grey pen (default 5).

-f or -file <output-file>
Write HPGL data to a file. By default the tool writes to standard output.

-nc / -nocolor
Do not generate color commands for the plotter. By default, pens 1,2,3 and 4 are used to show color.

-normalpen <pen-number>
This allows the user to set the 'normal' or black pen (default 1).

-pages <start>[-<end>]
Allows the user to select a group of pages to print. If you do not include a min or max page (e.g. “-5” or “10-”) it will assume the start or end, respectively, of the document.

-thinthick <number>
Sets the thickness that a thin line should be restroked to (if supported by plotter).

-thickthick <number>
Sets the thickness that a thick (inheritance) line should be restroked to (if supported by plotter).

-v Give more information about the conversion process. It is useful for tracking the progress through large schemas.

11.3 Overview

The **expg2hpgl** tool generates a stream of HP-GL drawing commands from an EXPRESS-G diagram. The generated HP-GL is similar to the display of **expgedit**, but lines will go through boxes, since HP-GL cannot erase areas. Also fonts on plotters tend to be limited. However, plotters allow for extremely large plots.

By default, the HP-GL is written to standard output. If this is not desired you can use either the **-file** or the redirect operator send the output to a file.

```
% expg2hpgl diagram.exg > outfile.hpgl
% expg2hpgl -file outfile.hpgl  diagram.exg
```

12 EXPRESS-G Set Options

12.1 Description

The EXPRESS-G options tool **expgsetopts** is a command line utility for changing page and font information in an EXPRESS-G diagram.

12.2 Command Line

expgsetopts [options] <expg-file>

The EXPRESS-G options tool accepts the following:

-help Print this list of options. The tool performs no other action and ignores all other options.

-fontname <name>
Set the font for printing. In order for **express2expg** to properly layout the boxes for printing it needs to have a description of the font. So if you change this after generating the document, be prepared for your text not to fit in the boxes.

-fontsize <pt. size>
Set the font size for printing. The default is 7 pt text. This can be too small for some purposes, but is quite readable, and reproducible on a 300dpi Postscript printer. Boxes are sized when the document

is laid out. If you change the font size it is quite possible that your boxes will no longer properly enclose the text.

-l / -landscape
Rotate the output 90 degrees on the page. This often results in using the longer dimension for the horizontal.

-p / -portrait
Rotate the output so that the long so that the longer dimension is vertical.

-pageheight <height-in-inches>
Set the size in inches of the height of the page. The default is 10.0". See **-pagewidth** for some usage notes.

-pagewidth <width-in-inches>
Set the size in inches of the width of the page. The default is 7.5". If you have a printer that takes larger paper you may wish to change this. You may need to leave a margin around the page where the printer is incapable of printing.

-v Give more information about the conversion process. It is useful for tracking the progress through large schemas.

12.3 Overview

The **expgsetopts** tool was originally intended to help migrate pre-v1.3 diagrams to the v1.4 release. When using the old package it was necessary to tell the tools what the page width, and height, and scaling factor to use at every invocation. With the new version this information is input once, when creating the EXPRESS-G file with **express2expg**. This tool is still useful for changing this information from the command line.

```
% expgsetopts -pageheight 10 -pagewidth 7.5 \
       -fontname Courier -fontSize 7 part-42.exg
```

Part Three: Conformance and Editing Tools

13 General STEP Conformance Checker

13.1 Description

EXPRESS information models contain rules and constraints that that applications can use to test data sets for correctness. This prevents the propagation of incorrect data from one application to another.

The STEP Conformance Checker can evaluate EXPRESS rules and constraints to verify databases defined by the STEP application protocols. It can examine every object in the database and determine whether it complies with the rules and constraints defined in the application protocol EXPRESS model.

13.2 Command Line

apconform [options] <data-file>

If no options are selected, the tool will perform all constraint checks. If options are provided, the tool will only perform the requested checks. The options accepted by **apconform** are:

-help Print this list of options and exit.

-all Verify everything. This is the default behavior.

-bounds Check aggregate bounds.

-unique Check uniqueness rules.

-inverse Check inverse attributes.

-atttypes Check attribute types.

-aggruni Check aggregate uniqueness constraints.

-arrnotopt Check for mandatory array elements.

-strwidth Check string widths.

-binwidth Check binary width.

-quiet Report only constraints that fail. Without this option, the tool also reports constraints that are unknown.

-realprec Check real number precision.

-refdom Check attribute reference domain.

-rules Evaluate all of the global rules.

-rule <name>
Evaluate the global rule called **<name>**.

-required Check required attributes.

-syntax Only check the syntax of the exchange file. Syntax checking is always done, but specifying this option turns off any other checks.

-type <name>
Check only entities of type **<name>**.

-verbose Report the status of all constraints whether passed, failed or unknown. Without this option, the tool does not print messages for constraints that pass.

-where Check entity where rules.

13.3 Windows Control Panel

The STEP Checker Windows control panel is shown in Figure 13.1. Run this by selecting **General AP Checker** in the ST-Developer Launcher or the “ST-Developer

Tools" Start Menu folder. The following sections describe the fields and setting on this control panel and show how to perform various tasks with the checker.

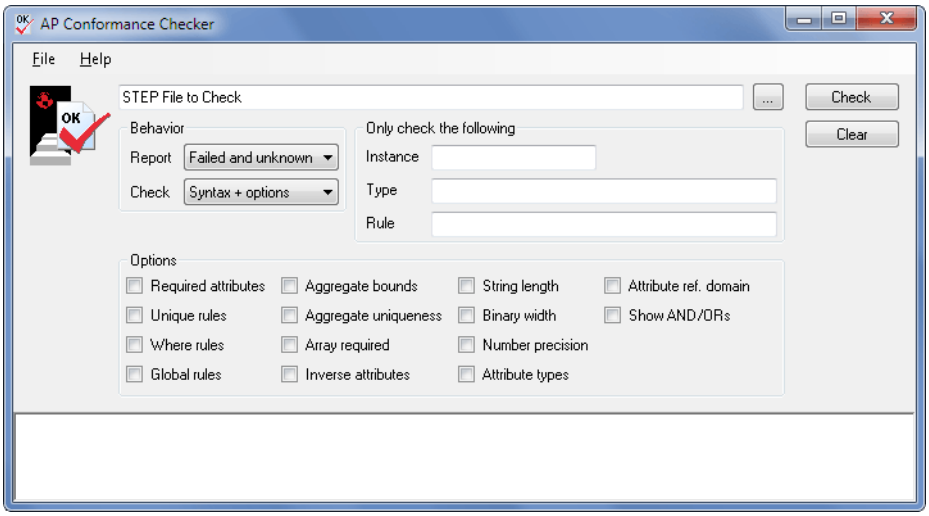


Figure 13.1 — AP Conformance Checker Control Panel

13.4 How to Check STEP Files

Since this checker is based on an EXPRESS interpreter, the checking process may take some time, depending on file size and constraint complexity. If your data files are based on AP203, AP209, or AP214 the specialized checkers instead will do the job quicker and also check extra constraints, like STEPnet vendor agreements and recommended practices. See **AP203, AP209 and AP214 Conformance Checkers** (Chapter 14, pp. 113) for details.

Otherwise, the General STEP Conformance Checker is the most appropriate tool. The conformance checker needs the data dictionary files (**<schema>.rose**) and pre-compiled parse data files (**<schema>_EXPX.rose**) for your EXPRESS schema.

If these files are not present in the ST-Runtime support files, you must generate them. Use the EXPRESS compiler control panel or call the compiler on the command line:

```
% expfront -rose [options] <express_file_name>
```

See **Generate Data Dictionary and Precompiled Schemas** (Section 2.7, pp. 22) for more information. For best results, copy the dictionary and parse data files to the ST-Runtime schemas directory so that they are available to all applications. Now you can invoke the testing tool:

13.4.1 Checking File Syntax

The first time you process a file from a new exchange partner, you should start by just verifying the file syntax. This quickly identifies larger issues such as missing schemas or malformed records without trying to do more detailed analysis.

At the top of the General AP Checker control panel shown in Figure 13.1 is a text field for the STEP files that you would like to check. You can open the file dialog using **Ctrl+O** or the **[...]** button to the right. You can also drag and drop files from the Windows Explorer.

In the **Behavior** area, select the **Syntax only** check option in the and press the **Check** button to start. From the command line, call the checker with the **-syntax** option:

```
% apconform –syntax datafile.stp
```

Once you have made sure that the Part 21exchange file is free of any basic formatting errors, you can perform more detailed analysis of the contents.

13.4.2 Checking Data Sets in Detail

Once you are satisfied that your data is syntactically correct, you can start checking the EXPRESS rules and constraints. You can check everything at once, which may take some time, or you can select individual constraints to check.

From the command line, call the checker as shown below. If you specify some options, the tool will perform just those checks, otherwise it will perform all possible checks.

```
% apconform [options] datafile.stp
```

On the Windows control panel, select **Syntax + options** box in the **Check** menu, then select any combination of constraint checks in the area below.

The checker prints the status of constraints as they are checked. You can control this with the **Report** options. The **Everything** option prints the status of each constraint as it is checked. The **Failed and unknown** and **Failed only** options print messages only if the constraint evaluates to the desired value.

If you are testing the output from one of your own applications, you will probably be concerned with specific entity types or constraints. You can fine tune the constraints and types that the tool will examine with the following options.

The **Required attributes** option checks whether each attribute either has a value or has been declared optional by the schema. On the command line, this is controlled by the **-required** option.

The **Unique rules** option evaluates the local "unique" rules specified as part of entity definitions. On the command line, this is controlled by the **-unique** option.

The **Where rules** option evaluates the local "where" rules specified as part of entity definitions. On the command line, this is controlled by the **-where** option.

The **Global rules** option evaluates all of the global rules specified in the EXPRESS. On the command line, this is controlled by the **-rules** option. You can also use the **Only this rule** field to select an individual rule for checking. On the command line, you can specify the **-rule** option with a rule name.

The **Aggregate bounds** option checks the size of aggregates against the upper and lower limits specified in the EXPRESS. On the command line, this is controlled by the **-bounds** option.

The **Aggregate uniqueness** option checks that sets and any other aggregates declared to have unique elements do not contain any duplicates. On the command line, this is controlled by the **-aggruni** option.

The **Array required** option checks whether all arrays elements either have values or have been declared optional by the schema. On the command line, this is controlled by the **-arrnotopt** option.

The **Inverse attributes** option evaluates inverse attributes. On the command line, these are controlled by the **-inverse** options.

You can use the **Only this type** field to select an individual entity type for checking. On the command line, you can specify the **-type** option with a type name.

The **String length** option checks the values of string attributes against any length constraints specified in the EXPRESS. On the command line, this is controlled by the **-strwidth** option.

The **Binary width** option checks the values of binary attributes against any length constraints specified in the EXPRESS. On the command line, this is controlled by the **-binwidth** option.

The **Number precision** option checks the values of REAL attributes against any precision constraints specified in the EXPRESS. On the command line, this is controlled by the **-realprec** option.

The **Attribute reference domain** and **Attribute types** options rerun some syntax

checks that are also performed when the data set is first read into the tool. On the command line, these are controlled by the **-refdom** and the **-atttypes** options.

The **Show AND/ORs** option prints a report listing the complex type found in the file. This can be useful when constructing a workingset file to generate C++ classes for application programming. On the command line, this is controlled by the **-andors** option.

14 AP203, AP209 and AP214 Conformance Checkers

14.1 Description

The **ap203check**, **ap209check** and **ap214check** tools rapidly check data files against the local rules (WHERE clauses) and global rules (RULEs) defined in the AP203, AP209 and AP214 EXPRESS schemas. The tool also checks for the presence of all required (non-OPTIONAL) attribute values in the data file.

The **ap203check**, **ap209check** and **ap214check** tools are faster than **apconform**, but are specific to a particular schema. Whereas the **apconform** tool utilizes a general EXPRESS interpreter, **ap2XXcheck** tools use custom C++ optimized for speed with techniques such as pre-computing reverse (USEDIN/INVERSE) references between entity instances.

14.2 Command Line

ap203check [options] <data-file>

ap209check [options] <data-file>

ap214check [options] <data-file>

The options accepted by the checkers are:

-help Print this list of options. The tool performs no other action and ig-

nores all other options.

14.3 Windows Control Panel

The AP203 Checker Windows control panel is shown in Figure 14.1. Run this by selecting **AP203 Checker**, **AP209 Checker**, or **AP214 Checker** in the ST-Developer Launcher or the "ST-Developer Tools" Start Menu folder. The following sections describe the fields and setting on this control panel and show how to perform various tasks with the checker.

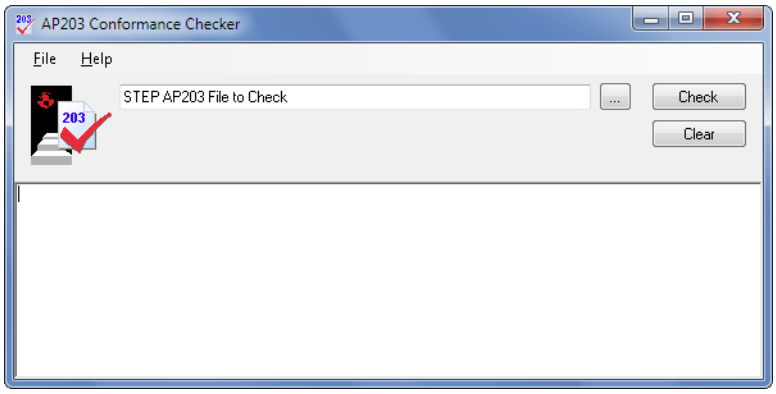


Figure 14.1 — AP203 Checker Control Panel

At the top of the AP203 Checker control panel is a text field for the STEP files that you would like to check. You can open the file dialog using **Ctrl+O** or the **[...]** button to the right. You can also drag and drop files from the Windows Explorer. Click the **Check** button to start checking.

14.4 Example

As an example, we test a file to see if it meets the AP203 schema constraints. The example file is an assembly of parts forming a doghouse. It is a modified copy of the file **doghouse.stp** on our web site translation service. It has been modified so that the geometric context entity instance specifies 2-dimensionality (while the geometry remains 3-dimensional) and so that one of the edge_loop instances does not meet a head-to-tail constraint. You can find this file in the ST-Developer installation as **demos/ap203check/dogbad.stp**

```
% ap203check dogbad.stp
== STEP AP-203 Checker
==
== Copyright (c) 1991-2006 by
== STEP Tools Inc., Troy, New York
== All Rights Reserved
==
== Product licensed to:
==      STEP Tools Internal Use Only
==      STEP Tools Lab
==

computing reverse pointers
...........
1125 total entity instances
validating required attributes
validating ENTITY advanced_brep_shape_representation, 1 instance
validating ENTITY advanced_face, 32 instances
WHERE failure: advanced_face WR5: edge_geometry(#505=intersection_curve) is
not one of line, conic, polyline, pcurve, b_spline_curve
  #483=advanced_face
WHERE failure: advanced_face WR5: edge_geometry(#500=intersection_curve) is
not one of line, conic, polyline, pcurve, b_spline_curve
  #483=advanced_face
WHERE failure: advanced_face WR5: edge_geometry(#495=intersection_curve) is
not one of line, conic, polyline, pcurve, b_spline_curve
  #483=advanced_face
WHERE failure: advanced_face WR5: edge_geometry(#488=intersection_curve) is
not one of line, conic, polyline, pcurve, b_spline_curve
  #483=advanced_face
WHERE failure: advanced_face WR5: edge_geometry(#488=intersection_curve) is
not one of line, conic, polyline, pcurve, b_spline_curve
  #483=advanced_face
WHERE failure: advanced_face WR5: edge_geometry(#604=intersection_curve) is
not one of line, conic, polyline, pcurve, b_spline_curve
  #506=advanced_face
WHERE failure: advanced_face WR5: edge_geometry(#599=intersection_curve) is
not one of line, conic, polyline, pcurve, b_spline_curve
  #506=advanced_face
WHERE failure: advanced_face WR5: edge_geometry(#594=intersection_curve) is
not one of line, conic, polyline, pcurve, b_spline_curve
  #506=advanced_face
WHERE failure: advanced_face WR5: edge_geometry(#589=intersection_curve) is
not one of line, conic, polyline, pcurve, b_spline_curve
  #506=advanced_face
WHERE failure: advanced_face WR5: edge_geometry(#584=intersection_curve) is
not one of line, conic, polyline, pcurve, b_spline_curve
  #506=advanced_face
WHERE failure: advanced_face WR5: edge_geometry(#579=intersection_curve) is
not one of line, conic, polyline, pcurve, b_spline_curve
  #506=advanced_face
WHERE failure: advanced_face WR5: edge_geometry(#574=intersection_curve) is
not one of line, conic, polyline, pcurve, b_spline_curve
  #506=advanced_face
```

```
WHERE failure: advanced_face WR5: edge_geometry(#569=intersection_curve) is
not one of line, conic, polyline, pcurve, b_spline_curve
  #506=advanced_face
WHERE failure: advanced_face WR5: edge_geometry(#564=intersection_curve) is
not one of line, conic, polyline, pcurve, b_spline_curve
  #506=advanced_face
WHERE failure: advanced_face WR5: edge_geometry(#559=intersection_curve) is
not one of line, conic, polyline, pcurve, b_spline_curve
  #506=advanced_face
WHERE failure: advanced_face WR5: edge_geometry(#554=intersection_curve) is
not one of line, conic, polyline, pcurve, b_spline_curve
  #506=advanced_face
[... 159 more messages]

validating ENTITY axis2_placement_3d, 41 instances
WHERE failure: axis2_placement_3d WR1: 'location' is not 3-dimensional
  #14=axis2_placement_3d
WHERE failure: axis2_placement_3d WR2: 'axis' is not 3-dimensional
  #14=axis2_placement_3d
WHERE failure: axis2_placement_3d WR3: 'ref_direction' is not 3-dimensional
  #14=axis2_placement_3d
WHERE failure: axis2_placement_3d WR4: could not compute cross product of
'axis' and 'ref_direction'
  #14=axis2_placement_3d
WHERE failure: axis2_placement_3d WR1: 'location' is not 3-dimensional
  #18=axis2_placement_3d
WHERE failure: axis2_placement_3d WR2: 'axis' is not 3-dimensional
  #18=axis2_placement_3d
WHERE failure: axis2_placement_3d WR1: 'location' is not 3-dimensional
  #26=axis2_placement_3d
WHERE failure: axis2_placement_3d WR2: 'axis' is not 3-dimensional
  #26=axis2_placement_3d
WHERE failure: axis2_placement_3d WR1: 'location' is not 3-dimensional
  #28=axis2_placement_3d
WHERE failure: axis2_placement_3d WR2: 'axis' is not 3-dimensional
  #28=axis2_placement_3d
WHERE failure: axis2_placement_3d WR3: 'ref_direction' is not 3-dimensional
  #28=axis2_placement_3d
WHERE failure: axis2_placement_3d WR4: could not compute cross product of
'axis' and 'ref_direction'
  #28=axis2_placement_3d
WHERE failure: axis2_placement_3d WR1: 'location' is not 3-dimensional
  #35=axis2_placement_3d
WHERE failure: axis2_placement_3d WR2: 'axis' is not 3-dimensional
  #35=axis2_placement_3d
WHERE failure: axis2_placement_3d WR1: 'location' is not 3-dimensional
  #43=axis2_placement_3d
WHERE failure: axis2_placement_3d WR2: 'axis' is not 3-dimensional
  #43=axis2_placement_3d
[... 98 more messages]

validating ENTITY calendar_date, 5 instances
validating ENTITY cc_design_date_and_time_assignment, 2 instances
validating ENTITY cc_design_person_and_organization_assignment, 5 instances
```

```
validating ENTITY design_context, 1 instance
validating ENTITY direction, 130 instances
validating ENTITY edge_loop, 33 instances
validating ENTITY face, 32 instances
validating ENTITY geometric_representation_item, 794 instances
validating ENTITY intersection_curve, 87 instances
validating ENTITY length_measure_with_unit, 1 instance
validating ENTITY length_unit, 2 instances
validating ENTITY local_time, 5 instances
validating ENTITY line, 73 instances
validating ENTITY measure_with_unit, 3 instances
validating ENTITY mechanical_context, 1 instance
validating ENTITY oriented_edge, 174 instances
validating ENTITY path, 33 instances
WHERE failure: path WR1: path is not head-to-tail (bad edge order?)
  #485=edge_loop
validating ENTITY person, 3 instances
validating ENTITY plane_angle_measure_with_unit, 1 instance
validating ENTITY plane_angle_unit, 2 instances
validating ENTITY product_definition_shape, 1 instance
validating ENTITY representation_item, 1035 instances
```

15 The STEP Conformance Editor

15.1 Description

The STEP Conformance Editor is a hypertext tool that can be used to create, view, explore, and edit STEP data files, and with its built in EXPRESS interpreter can interactively evaluate constraints, global rules and derived attributes.

This tool is an X11 application that is currently only available with the UNIX version of ST-Developer.

15.2 Command Line

stepedit [options] [data-file]

The tool recognizes the options shown below as well as the standard set of X11 command line options.

-help Print this list of options. The tool performs no other action and ignores all other options.

-nocheck Disable the EXPRESS interpreter. Under this option, only the explicit attributes will appear in the panes. The tool will not try to display or evaluate constraints or derived attributes.

-panes <number>
Controls how many display panes the editor initially has. The default is two. Panes can also be added or removed with the buttons at the bottom of the window.

-trace Turn on printing of the EXPRESS interpreter execution log. This causes detailed tracing of the flow of control through the WHERE rules, and functions. Use this if the tool seems to be having trouble with a particular construct. This option in only available for **stepedit.**

15.3 Starting the Editor

The STEP editor must be started on an X11 display. Be sure that your **$DISPLAY** environment variable is set correctly.

The editor needs the data dictionary files (**<schema>.rose**) and precompiled parse data files (**<schema>_EXPX.rose**) for your EXPRESS schema.

If these files are not present in the ST-Developer **systems_db/schemas** area, you must generate them. Use the EXPRESS compiler control panel or call the compiler on the command line:

```
% expfront -rose [options] <express_file_name>
```

See **Generate Data Dictionary and Precompiled Schemas** (Section 2.7, pp. 22) for more information. For best results, copy the dictionary and parse data files to the ST-Developer **systems_db/schemas** area so that they are available to all applications. Now you can invoke the editor:

```
% stepedit
```

To view a particular STEP file, use the command:

```
% stepedit <data-file>
```

If the STEP file is in the search path, the editor will read this file into memory to start the session.

15.4 Visual Layout

The editor displays STEP objects as formatted text, using the X Window System to implement a mouse-driven user interface. Most of the editor operations are invoked by clicking on buttons or pieces of text, called *action tags*, within the editor display.

The main editor window contains several panes, each of which can display a single STEP object. As you move through a STEP model, the editor keeps a trail of objects you have descended through. The panes act as a window onto this list. Each pane has horizontal and vertical scroll bars for objects which are too large to fit entirely within the pane. Beneath the panes are buttons which perform additional functions.

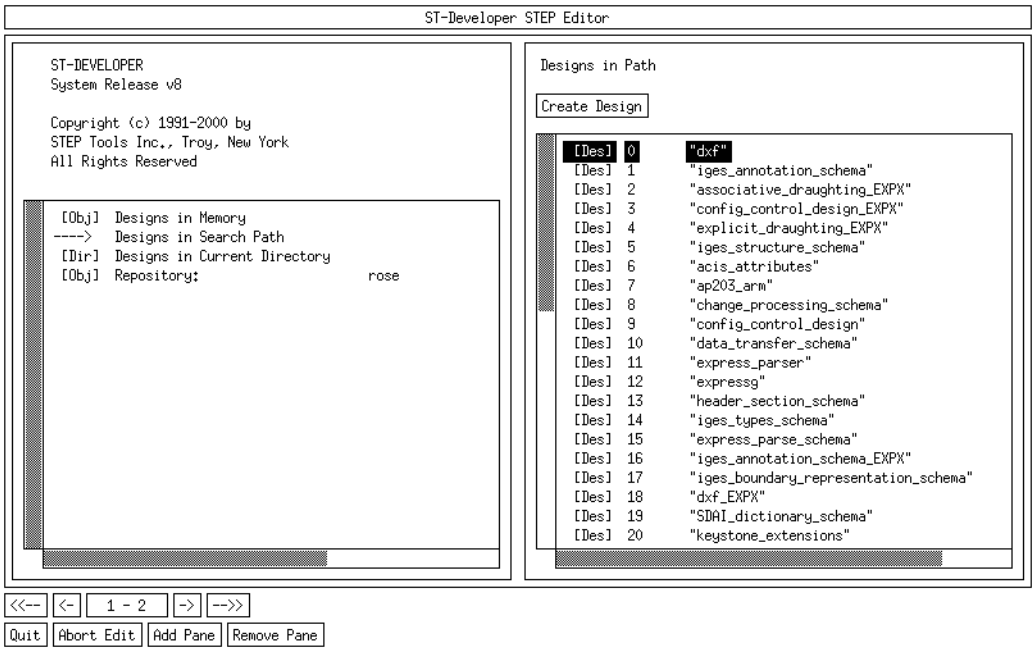


Figure 15.1 — Editor Main Window

15.4.1 Message Bar

At the top of the main editor window, you will notice a message bar. This will display instructions and error messages where appropriate. If the editor seems to be acting strangely, or if it beeps, check the message bar to see if there is an error message.

15.4.2 Scroll Bars

To use the scroll bars, click the left mouse button to move forward, the right to move backward, and the center to “pick up” the indicator and move it to a particular spot.

15.4.3 Buttons

As you move through a STEP model, the editor keeps track of the objects you visit. In effect, this list is the path that you followed to get to your current position. The panes act as a viewing window on this sequence of objects. The arrow buttons may be used to position the viewing window on this sequence.

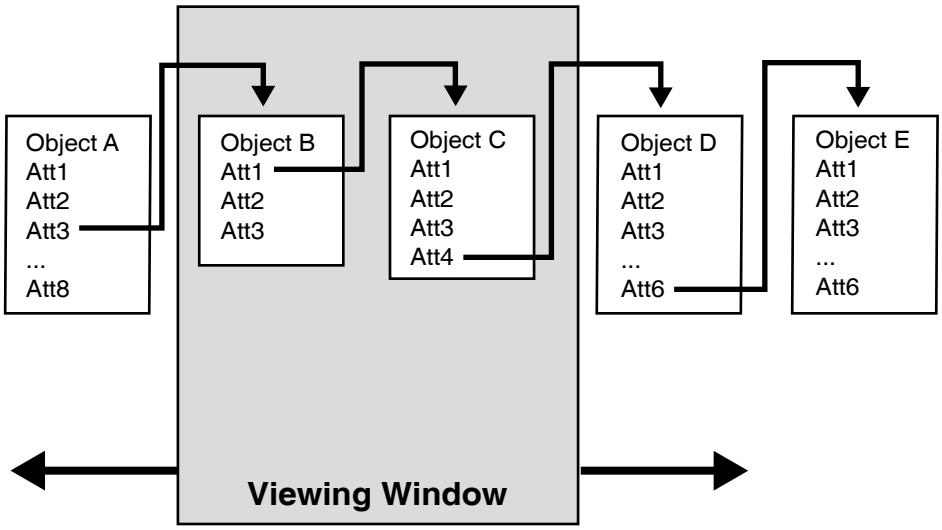


Figure 15.2 — Viewing Window on the Object Path

<-<- Moves the viewing window to the start of the list. This is the pane that displays the list of designs. This can also be accomplished by **Ctrl+‘<’**.

<- Moves the viewing window one object to the left, if possible. This may also be done with ‘**<**’

-> Moves the viewing window one object to the right, if possible. Anytime that a new object is selected, all panes beyond the current one are replaced. This may also be accomplished by typing ‘**>**’

->-> Moves the viewing window to the last object in the list. This may also be accomplished by typing **Ctrl+‘>’**

Pressing the arrow keys moves the highlighted area around the display. Pressing a left or right arrow key moves the highlight to the next or previous panel. If the panel is the last visible in the window, the window is scrolled as if the **<-** or **->** button had been clicked. The full set of navigation keys is:

<Shift+←> Shift all panes one slot to the left. (Same as the **<-** button).

<Shift+ →> Shift all panes one slot to the right.

<←> Move the highlight one pane to the left, scrolling if necessary.

<→> Move the highlight one pane to the right, scrolling is necessary.

<↑> Move the highlight one line up

<↓> Move the highlight one line down.

<Tab> Toggle between the top and bottom sections of a pane.

In addition to the arrow buttons, there are a number of other buttons or indicators that you will find at the bottom of the main editor window:

Panes The Panes "button" has no effect, but shows your current position within the list of objects. The **add pane** and **remove pane** buttons allow you to customize the appearance of the editor. By adding a pane, more objects can be viewed, by removing a pane, the need to scroll can be minimized. The '**+**' and '**-**' keys may also be used to add and remove panes.

Quit The quit button will dismiss the editor. A dialog box will verify that you really want to quit. You may also use the **<Q>** key for the same action.

Abort Edit The abort edit button will stop an edit, delete, or insert operation (text box on screen) as will the **<Esc>** key, or a mouse click elsewhere in the window.

15.4.4 Object Display Panes

The format of a display pane varies across different types of objects. A complete description of each different type can be found in the final section. All panes, however, have the following elements in common:

- A header, containing common information. For example, the header for STEP objects contains the object's name, design, domain, and object identifier.

- Action tags, such as **[save]** and **[insert]**, that are appropriate to the object. These are located below the header, and generally appear only if the object is writable. Clicking on the tag will perform the action.

- Instance variables contained by the object. These entries are arranged in three columns. The rightmost column contains the data values, the middle column contains the attribute name or index within an aggregate, and the leftmost column contains navigation tags, if any are appropriate.

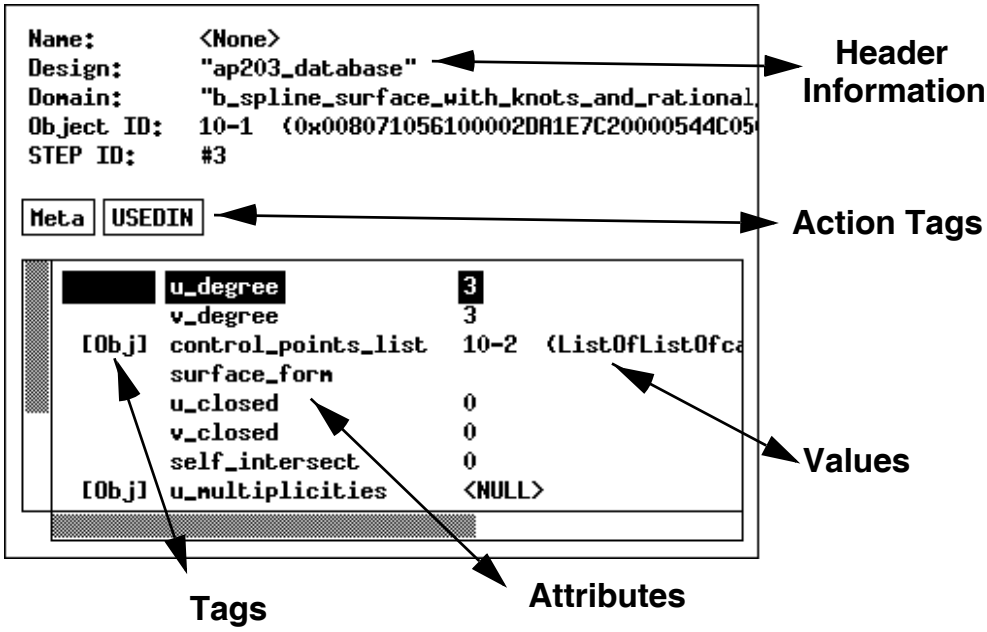


Figure 15.3 — Typical Object Display Pane

Headers

The top section of each object display pane provides general information about the STEP object. The fields you will see are:

Name Each design keeps a list of named objects. If an object has been given a name, the editor will always refer to it by name instead of object identifier. This field shows the object's name, if it has been given one. To give an object a name, or to change an existing name, edit this field. To remove an object's name, set this field to **<NULL>**.

Domain Each object has a particular type associated with it. The domain field shows the object's type. An object can be mutated to a new type by changing this field. However, this may violate constraints on the data model, so this option should be used with care. It is not possible

to mutate aggregates.

Permission The permissions field identifies the object as either writable or read-only. This field may not be changed.

Object Identifier (OID)
The OID field contains the unique identifier for the object — its OID. Two values are given here: the short OID, and the long OID. The short OID is simply a shorthand form of the long OID and is only valid this session, while the long OID will always be valid, in any session.

STEP File ID
This is the entity ID of the STEP file. This field only appears on entities (RoseStructures), and only when the data was read in from a STEP Part 21 file. This is useful when cross referencing data between the STEP editor and the Part 21 file.

Action Tags

Each different type of object may have a certain set of actions associated with it. Generally these are made available to you as "action tags" — selectable pieces of text within a display pane. An action tag can be recognized by the convention of enclosing the text within brackets. Clicking on the tag with the mouse will cause the action to be performed. All action tags have keyboard shortcuts, which are given after the tag's name.

[Create design] / <C>
Creates a new design in the workspace. You will be prompted for a design name. The design will be created, and displayed in the next pane. Found on lists of designs.

[Delete] /<D>
Removes an element from a list. Click on the element you wish to delete. Found on dictionary and aggregate panes.

[Delete object] /<D>
Deletes an object from the design. Click on the element you wish to delete. Found only on the "All Objects" pane of a design.

[Insert] /<I>
Add an element to a list. Click on the location within the list of elements. An edit box will appear for the new value. Found on dictionary and aggregate panes.

[Meta] /<T>
Display the compiled EXPRESS definition for the object in the next

pane. Found on the pane of any STEP object.

[Move] /<M>
Move an element within a list. Click the element to move and then on the new location within the list of elements. Found on aggregate panes.

[New object] /<N>
Create a new object in the design. Prompts you for the domain name. See Creating Objects for more information. Found only on the "All Objects" pane of a design.

[Save] /<S>
Save the design to secondary storage. Found on design panes only if the design is writable

[USEDIN] /<U>
Compute and displays a list of the objects that reference the displayed object in the same design. Found on the pane of any STEP object.

Instance Variables

At the bottom of each pane is a scrollable list of the instance variables associated with this object.

The first list may contain some "navigation tags". These indicate that the instance variable contains a reference to more information of some kind. Clicking on the tag, or pressing return when a line containing the tag is highlighted will move the focus to the information in question. The action tag indicates the type of information being offered. The tag **[OBJ]** means a STEP object, **[DES]** means design object, and **[DIR]** means a directory of files.

The second list usually shows attribute names. The display for aggregates contains an index. In SELECTS and dictionaries, this field can be changed by clicking on it, or by typing **<Shift+E>**.

The third list shows the value of each instance variable. This value may be changed by clicking on it, or pressing **<E>** on the keyboard. If the object is writable, an edit box will appear and you may type in the new value.

Objects are displayed by name if they have one, or by (short) object identifier. The type of an object is displayed after it's OID in parentheses. See the section **Primitive Data** (Section 15.6.4, pp. 135) for the display format of each type of data values.

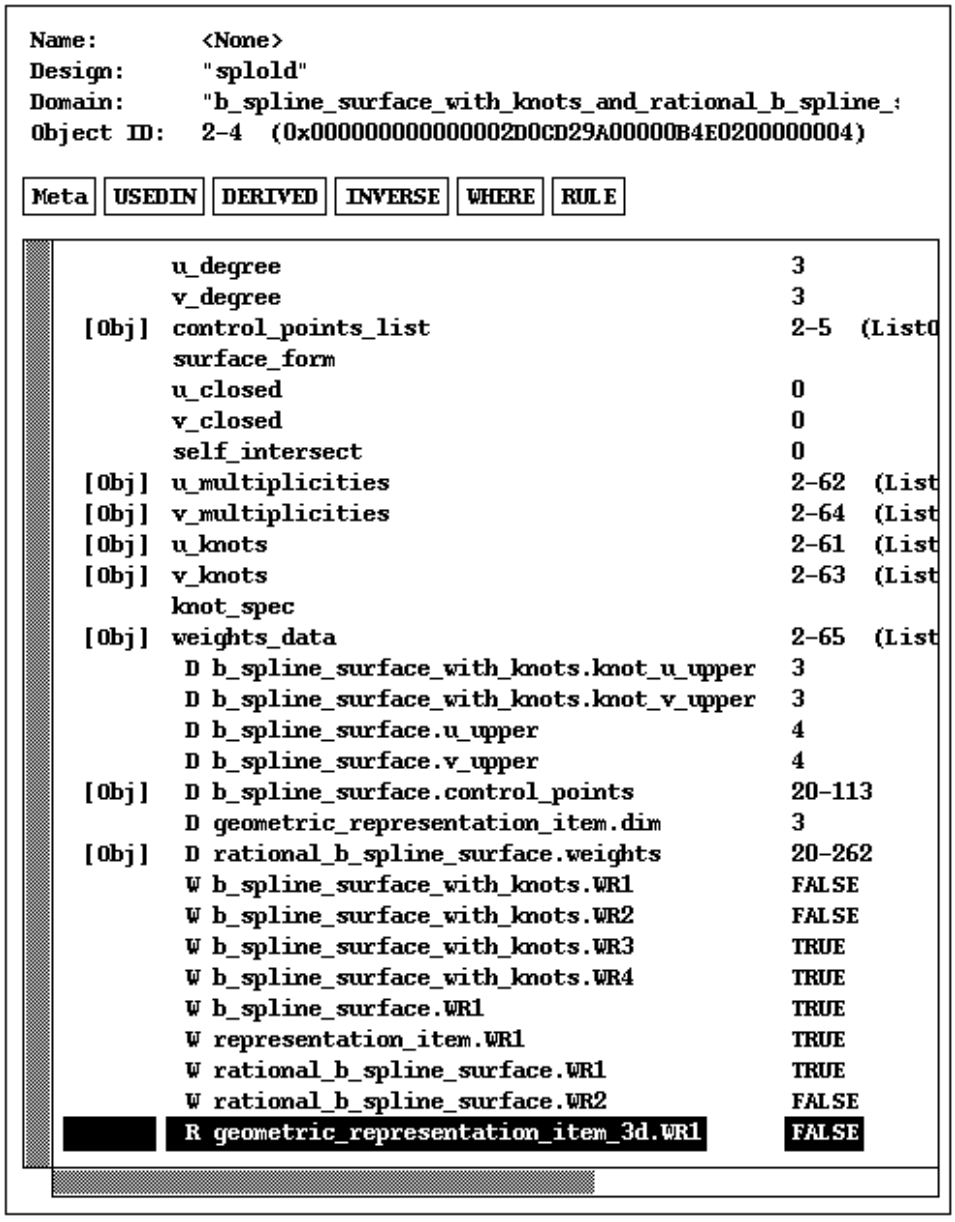


Figure 15.4 — Object with Constraints and Derived Attributes

15.4.5 Constraints and Derived Attributes

The **stepedit** tool displays entity instances with information about optional attributes, derived attributes, where rules, and global rules.

Each calculated attribute takes one line in the display. Each of these lines has the basic form of:

```
[*-][DIWR] <Attribute name> <Attribute Value>
```

The meanings are:

```
'*'   Optional attribute
'-'   Potentially needs updating
'D'   Derived attribute
'I'   Inverse attribute
'W'   Where rule
'R'   Global rule
'?'   Indeterminate value.
```

The tool will list, normal attributes followed by derived attributes, then Inverse attributes, then local where rules, and finally global rules.

For all of the calculated fields (derived attributes, inverses, where rules, and global rules), if the field is inherited, it will prefix the name of the field with the name of the entity that defined the field.

If the field was not given a name the interpreter will construct one for it (eg. 'WR1', 'WR2' ... for where rules).

Before a Calculated field is evaluated (by clicking in it's value column) it will have the value **<UNEVAL>**. If **stepedit** encounters an error while calculating the value it will display **<ERROR>**. If the value is indeterminate, or **stepedit** is unable to calculate the value, it will put a question mark as its value.

Evaluating Rules and Attributes

Initially, the rules and derived attributes appear as **<UNEVAL>**. This indicates that the value has not been computed. A minus sign preceding the "D", "I", "W" or "R" means that the value may be out-of-date, and should be recomputed.

To evaluate all rules at once, click on the **[WHERE]** action tag at the top of the pane. To compute all derived attributes, click on the **[DERIVED]** action tag (similarly for inverses, and global rules). To evaluate just an individual rule or attribute, click on the value column for that entry. Once evaluation is complete, the minus sign will disappear, or the **<UNEVAL>** will be replaced with the actual value.

Also note that you can look at the schema information for entities, and can also track down all objects that make reference to this object (among the loaded designs). These are accessed by clicking on the **[Meta]** and **[USEDIN]** buttons respectively.

Execution Log

The **stepedit** tool prints an execution log to standard output. This is a a trace of functions evaluated, return values, errors and warnings. The line number in the EXPRESS file where the problem occurred is included in the error or warning message. The execution log is not displayed for **stepedit_v2**.

15.5 Navigating

The editor uses a navigational interface. You move from object to object by clicking on the "navigation tags" described in the previous section. This section will focus on other the ways to get to objects using the search and index features of the editor.

The editor keeps a record of the most recent path taken through the hierarchy. This record includes the initial window, and the lowest object which you have visited in the hierarchy. Clicking on the buttons which contain double arrows automatically moves you to the window containing the highest (arrows pointing to the left), or lowest (arrows pointing to the right), object in the record. The single arrows move you up and down through the hierarchy one pane at a time. The center button performs no action, but simply indicates which panes are displayed in the window.

The first pane displayed by the editor contains three lists of designs. These lists store the designs already in memory, the designs in your search path, and the designs in the current directory.

Designs in Memory
: A list of the designs that have been loaded into memory by the editor. Initially this list simply contains some system-defined schemas, such as "keystone3_0" that are always present.

Designs in Search Path
: A list of the designs that can be found in the search path. If you select one of these designs, it will be read in from secondary storage. If the design is already in memory, it will not be re-loaded.

Designs in Directory
: A list of the designs that can be found in the current directory, as well as a list of subdirectories. You can find designs that are not in the search path, by navigating up and down through the directory hierarchy.

Use these lists to finds the design you want. The design will appear in the next pane

once it has been read in. This pane contains the search and index features that will let you find and move to the objects within the design.

All Entities
A list of all the entities (RoseStructures) in the design.

All Objects
A list of every object contained within the design. This includes all selects, aggregates, and other objects which may not exist on their own.

Some Objects
A search pane with fields for search criteria. This feature allows for searches based on name and/or type. This is useful for navigating in larger designs.

Named Objects
A list of every object which has been given a name.

STEP Name
A pane containing the name header information from the STEP file.

STEP Description
A pane containing the description of the STEP file. In a Part 21 file, this information is in the header.

STEP Schemas
A list of the compiled express schemas associated with the design. Before you can create an object of a particular type, the schema which defines the type must be added to this list.

Root Object
Some application programs will designate an object as "root," and then use this object as a starting place for navigating to other objects. Only ROSE working form file have a root object. All other files formats (e.g., STEP Part 21) have a NULL root object.

15.5.1 Finding Objects By Some Criteria

The "Some Objects" pane allows you to search for objects matching some search criteria. These criteria — a specific name, a domain, a STEP file entity ID, or a "where" clause based on the value of an attribute — will be matched against all objects in the design. Blank fields will be ignored. If you fill in the domain field, all objects of that type will be listed in the next pane. If the name field is filled in, the object with that name, if it exists, will be shown in the next pane. Once you have

filled in the field(s), hit **<return>** repeatedly or click on the “Select Objects” tag to start the search.

Here we see a search for all objects of type **Point**. The search has resulted in four objects.

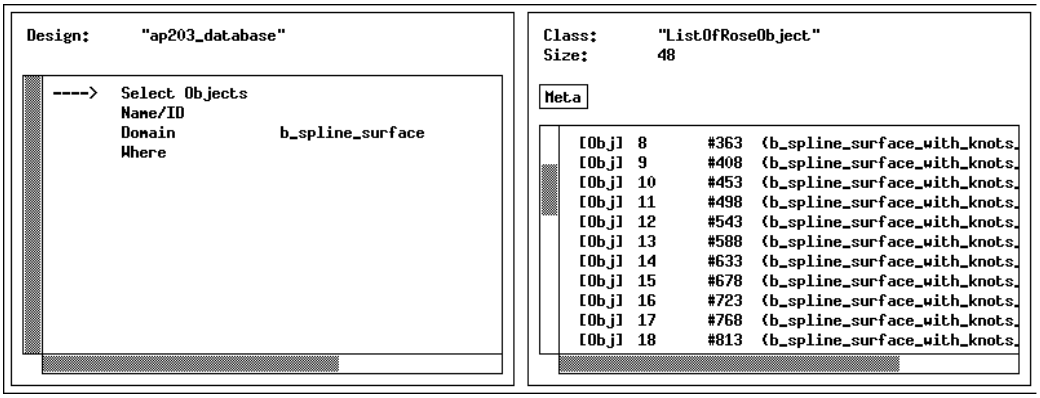


Figure 15.5 — Search By Type

And in the next example, we see a search for all objects named “**c1**.” The search has shown that there is a Circle object with that name. If we had also specified the domain “**Point**,” the search would have been unsuccessful, since the object named “**c1**” is a Circle.

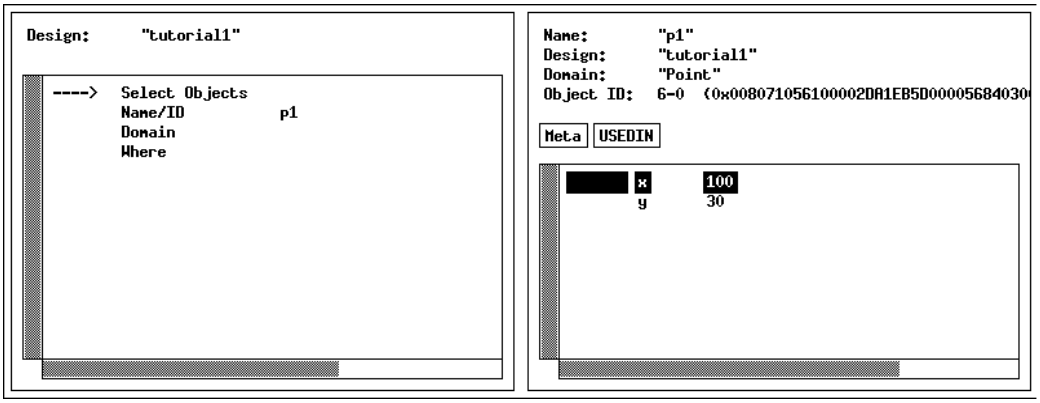


Figure 15.6 — Search By Name

Likewise, we can also search by STEP ID (line number). Please note that this feature only works for data that was read in from a STEP part 21 file. To do so, type

the ID number on the Name/ID line.

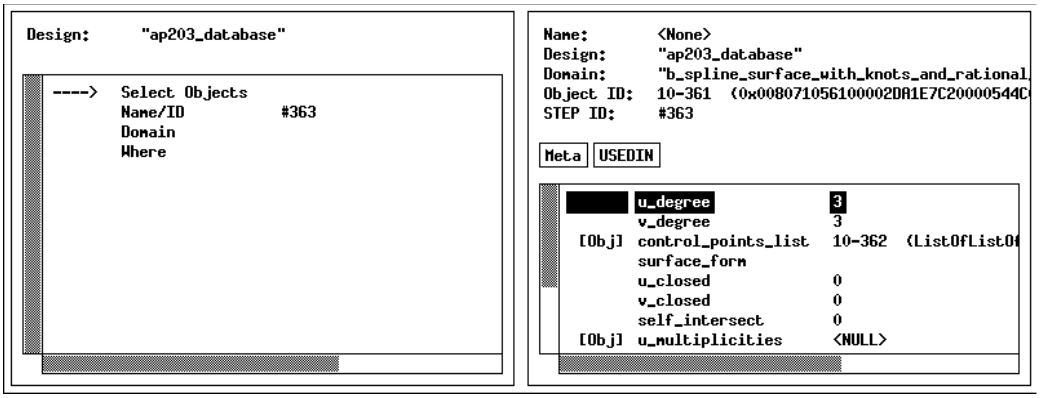


Figure 15.7 — Search By STEP ID Number

In addition to these criteria, the "**Some Objects**" pane, and several of the commands in the ROSE Database Utility, let you specify a "where clause" to select objects that match constraints based on attribute values and other criteria. Some examples of where clauses:

To select all Points whose X coordinate is less than or equal to 0.0:

```
domain='Point' and x<=0.0
```

To select all instances of objects whose entity definition is in the "topology" schema:

```
this.schema = 'topology'
```

In the previous example, the keyword "this" was optional. The same query could have also been written as:

```
schema='topology'
```

To select all Polygons if the X coordinate of the first vertex is not equal to 0:

```
domain='Polygon' and this.vertices[0].x <>  0.0
```

To select all objects in which the label attribute is "A little Picture":

```
label='A little Picture'
```

And in the next example, we see a search for all objects with an "**x**" attribute with value greater or equal to 1. The search has resulted in three Point objects.

The complete syntax for the where clause is:

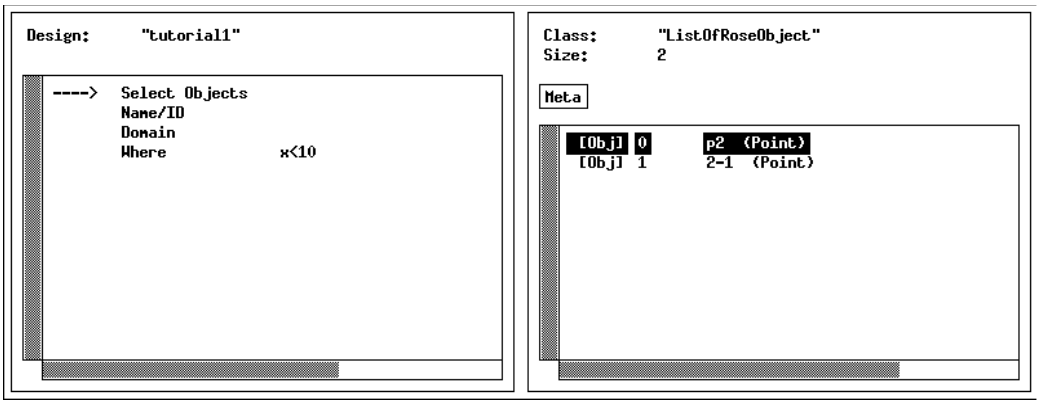


Figure 15.8 — Search By Value

```
<where-clause> = <expr>
<expr> =
        |       <term>
        |       <term> 'AND' <expr>
        |       <term> 'OR' <expr>
<term> =
        |       '(' <expr> ')'
        |       <operand> <rel-op> <operand>
<rel-op> = '<' | '>' | '=' | '<>' | '>=' | '<='
<operand> =
        |       <EXPRESS string>
        |       <integer>
        |       <REAL number>
        |       <attr> <recursive-attr-spec>
<attr> =
        |       'THIS' | 'DOMAIN' | 'DESIGN' | 'SCHEMA'
        |       <an attribute of the object being tested>
<recursive-attr-spec>
        |       '.' <attr> <recursive-attr-spec>
        |       '->' <attr> <recursive-attr-spec>
        |       '[' <integer> ']' <recursive-attr-spec>
        |       '[' <EXPRESS string> ']' <recursive-attr-spec>
        |       nil.
```

15.6 Editing

This section describes how you can use the editor to modify the attributes of a STEP object, to add new STEP objects, and to remove unwanted STEP objects.

15.6.1 Changing an Attribute Value

An attribute value may be changed by clicking on it. If the object is writable, an edit box will appear and you may type in the new value. Type **<return>** after you have entered the correct value. If you decide that you do not want to edit the value after all, you can click on the "**abort edit**" button at the bottom of the window or press the **<Esc>** key. To save the new value to secondary storage, click the **[save]** action tag, on the pane for the design.

If a design object is locked, no part of it may be edited. Some action tags that modify a design, such as **[new object]** and **[delete object]**, appear only if the design is writable.

The editor does not allow schemas to be changed. For this reason, RoseDomains, RoseAttributes, and lists thereof are read-only.

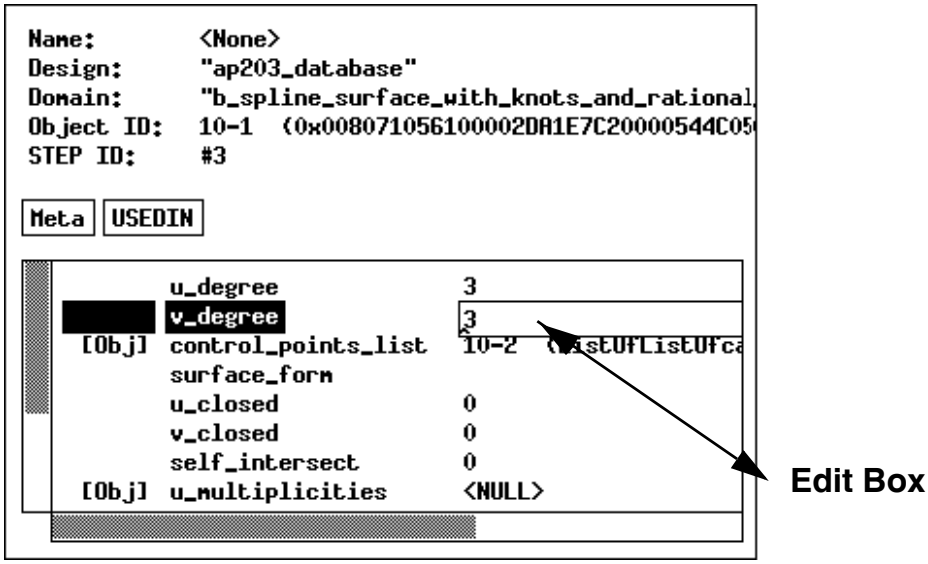


Figure 15.9 — Editing an Object

15.6.2 Saving Changes

To save your changes to secondary storage, click the **[save]** action tag, on the pane for the design.

15.6.3 Edit Box and Key Bindings

The edit box has a number of special key bindings. As discussed earlier, the **<Return>** key is used to accept a change, and the **<Esc>** key or the **abort edit** button can be used to bail out of a change. You can use the arrow keys to move the cursor back and forth. After an edit is completed, and there is no error, the box reappears in the next field that may be changed. The following Emacs-like key actions are also used:

```
Ctrl-A          beginning-of-line
Ctrl-E          end-of-line

Ctrl-B          backward-character
Ctrl-F          forward-character

Meta-B          backward-word
Meta-F          forward-word

Ctrl-D          delete-next-character
Meta-d          delete-next-word
Ctrl-H          delete-previous-character
Meta-h          delete-previous-word
Ctrl-K          kill-to-end-of-line
Ctrl-T          transpose-characters
Meta-D          kill-word

Meta-H          backward-kill-word
Meta-K          kill-to-end-of-paragraph

Shift Meta-BackSpace  backward-kill-word
Shift Meta-Delete     backward-kill-word
```

15.6.4 Primitive Data

The editor recognizes the following forms for data values. In addition to the following forms, the string “NULL” or “<NULL>” may be specified for any data value.

INTEGER Standard C integer formats.

REAL Standard C float formats.

STRING The string may optionally be surrounded by single or double quotes. If it is not, however, all leading and trailing whitespace is removed.

BOOLEAN Accepts **[t]rue, [f]alse, [y]es**, and **[n]o** Case insensitive.

LOGICAL Accepts **true**, **false**, **unknown**. Case insensitive.

BINARY Accepts a string of hexadecimal numbers.

ENUMERATION
Accepts enumeration values as strings.

Object attributes are slightly different than the primitive attributes. The object attributes contain references to instances rather than the instances themselves. The following forms are used to indicate object references:

<nn-nn> Refers to the ROSE Object with the shorthand object identifier <nn-nn> in the current design. e.g. **<1-25>**, **<12-156>**

<"name" nn-nn>
Refers to the ROSE Object by shorthand object identifier in a foreign design. e.g. **<"picture" 1-25>**, **<"wing" 2-87>**

"name" Refers to the object called "name" in the current design. The quotes are optional.

#nnn Refers to the object with STEP file line number "nnn." This option only makes sense for data read from STEP Part 21 files.

To fill a field with a new object, type the reserved name "**new**". The domain of the new object will be taken from the field type. If you want to create a more specific type of object, type "**new <domain-name>**". The following section gives more information about creating objects.

15.6.5 Creating Objects

The easiest way to create a new object is as an attribute of an existing object. Type in the reserved name "**new**" and a new object will be created to fill the attribute. The domain of the new object will be taken from the attribute type. If you want to create a more specific type of object, type "**new <domain-name>**". The newly-created object will be displayed in the next pane.

Objects can also be created from the "**All Objects**" pane of a design. There is a **[new object]** action tag. Click on this tag and an edit box will appear in the object list. Fill in the name of a domain. There is no need to type "**new.**" An object of that domain will be created and displayed in the next pane.

New Designs

New designs can be created from within the editor. If you go to any of the top-level lists of designs, you will find a **[new design]** action tag. Click on this tag and an edit box will pop up. Type in the name that you would like the new design to have. A design object with this name will be created and displayed in the next pane. If a design with that name already exists, no design will be created and an error message will be displayed.

Be sure to fill in the list of schemas with the names of the compiled EXPRESS schemas you would like to use. If you do not specify any schemas the editor will not be able to create any objects in the new design. If you tend to use a fixed set of schemas, you can set up a template so that the list will be automatically initialized.

Templates

Should you need to create large amounts of data with the editor, you can reduce the amount of repetitive data entry you have to do by using templates. You can use templates to set up default values for certain types of entities, and create a complete data set with less effort.

When you create an object, the editor will search for a template object of the same type or a supertype to initialize the new object with. All templates are held in a design called “TEMPLATE”. If a template is found, its instance variables are copied to the new object. If the template contains a reference to another template object, that object is also copied.

For example, when creating a Point, the editor will look at the Point template and copy the “x” and “y” coordinates to the new Point. When creating a Line, the editor will look at the Line template and make copies of the Point objects it finds there.

To find a template object, the editor will search the TEMPLATE design for an object with the same name as the new domain. If one does not exist, the editor looks for any object of the new type. For example, if you are creating a Point object, the editor will search the TEMPLATE design for an object named “Point”. If there is no such object, the editor looks for the first Point object it can find in the TEMPLATE design. If it still can't find a match, it will repeat the process with the supertypes of Point.

When you create a new design, the editor will copy the schema list from the TEMPLATE design. The editor searches for a TEMPLATE design at start-up. It can be created with the editor, or the “**create**” option of the ROSE data tools.

15.6.6 Deleting Objects

Objects can be deleted from the "**All Objects**" pane of a design. There is an action tag called **[delete object]**. Click on this tag and then click on the object you wish to delete. To cancel the operation press the "**abort edit**" button.

15.7 Specialized Panes

15.7.1 All Designs

The first pane in the editor presents the list of all designs. The copyright info is displayed at the top of the screen.

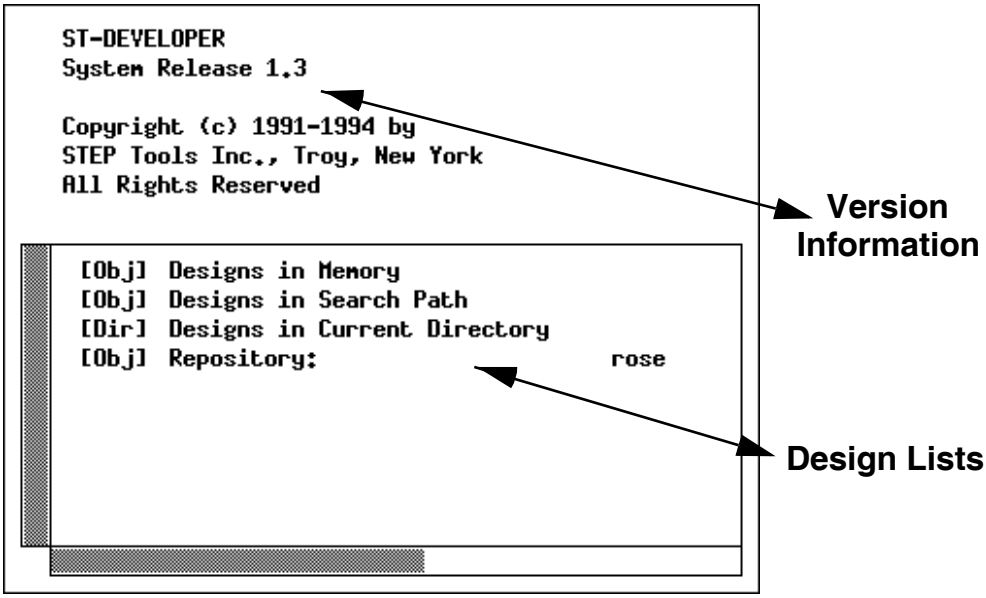


Figure 15.10 — All Designs Editor Pane

This pane contains three lists of designs:

Designs in memory
a list of the designs that have been loaded into memory by the editor. Initially this list simply contains some system-defined schemas, such as "keystone3_0" that are always present.

Designs in the search path
a list of the designs that can be found in the search path. If you select

one of these designs, it will be read in from secondary storage. If the design is already in memory, it will not be re-loaded.

Designs in a directory
a list of the designs that can be found in the current directory, as well as a list of subdirectories. You can find designs that are not in the search path, by navigating up and down through the directory hierarchy.

15.7.2 Design

The design pane offers several access routes to the objects contained within the design. This pane contains a lists of all objects and named objects, a pane to search for all objects of a particular type, and a place to designate a special “root” object for the design.

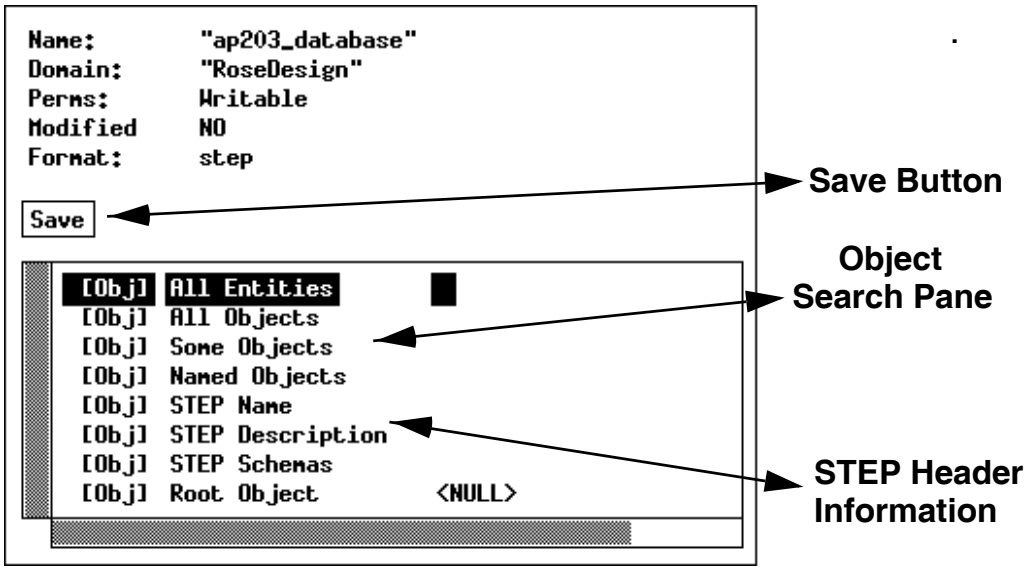


Figure 15.11 — Design Contents Editor Pane

All Entities
A list of all the entities (RoseStructures) in the design.

All Objects
A list of every object contained within the design. This includes all selects, aggregates, and other objects which may not exist on their own.

Some Objects
A search pane with fields for search criteria. This feature allows for

searches based on name and/or type. This is useful for navigating in larger designs.

Named Objects
A list of every object which has been given a name.

STEP Name
A pane containing the name header information from the STEP file.

STEP Description
A pane containing the description of the STEP file. In a Part 21 file, this information is in the header.

STEP Schemas
A list of the compiled express schemas associated with the design. Before you can create an object of a particular type, the schema which defines the type must be added to this list.

Root Object
Some application programs will designate an object as "root," and then use this object as a starting place for navigating to other objects. Note that only ROSE working form file have a root object. All other files formats (e.g., STEP Part 21) have a NULL root object.

To save the changes to a design, click the **[save]** action tag. If you quit the editor without saving all your designs, you are given an opportunity to save everything at that time.

15.7.3 Entity

The simplest type of STEP object is the entity. This consists of a number of named fields. The values of each field may be changed.

15.7.4 Select

A select has only one field, but the type and value of the field may both be changed.

15.7.5 Aggregates

An aggregate is a collection of elements of similar type. The four types of aggregates have quite different semantics. Bags are unordered collections; sets are unor-

dered collections without duplicates; lists are ordered collections; and arrays are fixed size collections.

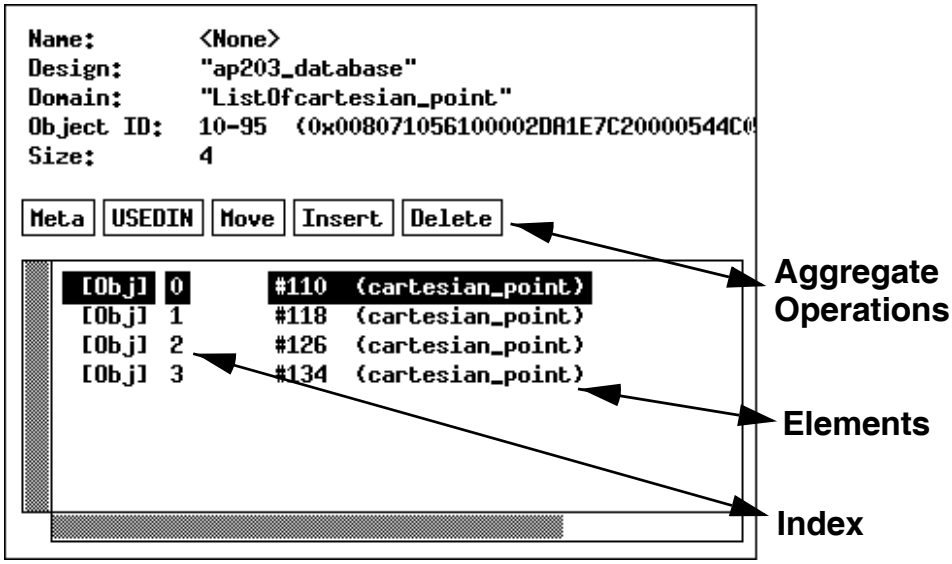


Figure 15.12 — Aggregate Contents Editor Pane

List, Bag and Set

To insert a new value into a set bag or list, click on the **[insert]** tag. Bags and sets are not ordered and will append the new value. An edit box will appear after the last element in the aggregate. Lists are ordered, and so you must click on the place where you would like the new value to go. Type in the new value, and then press **<return>**. You may also click on the bottom of the list (after the last element), and an edit box will appear. After an insert is completed, the next line is always selected for editing. To quit, press **<Esc>**, or **abort edit**.

To delete an element from a list, click on the **[delete]** tag, then on the item to be removed. The item will be removed from the list.

Array

Arrays are fixed size so the insert operation does not apply. Instead of inserting data into an array, you must change the size of the array and then modify the existing elements. To do so, change the “size” field in the header to reflect the new size. The array will be resized and any new elements will be set to <null>.

15.7.6 Dictionaries

A dictionary is a lookup table between a string and a value. The value can be any type known to ROSE. The list of named objects in a design is a dictionary, so the methods detailed here will also work there.

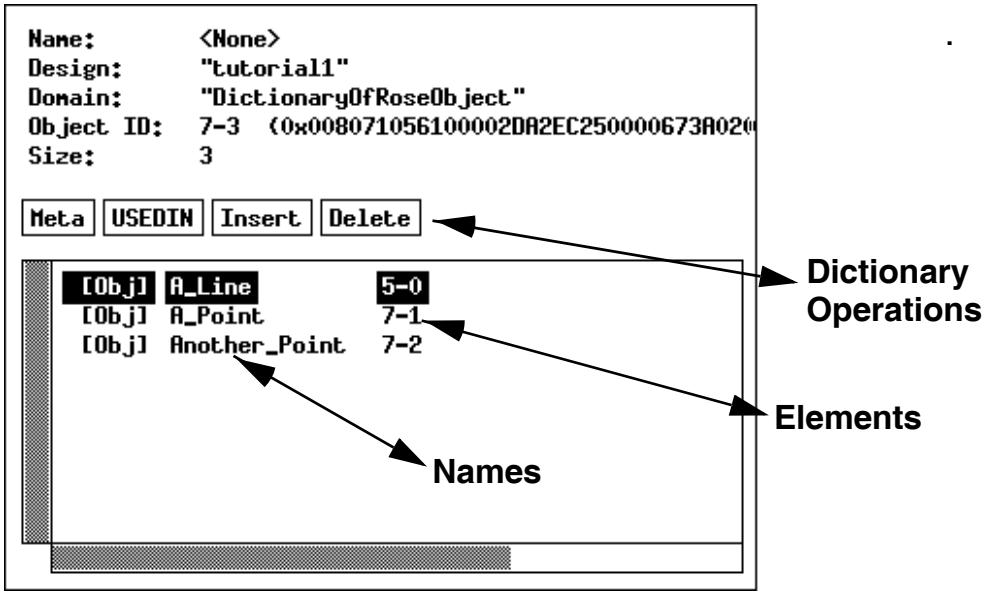


Figure 15.13 — Dictionary Contents Editor Pane

A key can be changed, so long as it is unique. To change a name, click on the center list and type a new value.

To add a new value to a dictionary, click on the insert button, then select the place in the dictionary to insert the object. Now, type the name followed by the value. If either is incorrect, it will be rejected with an error message.

To delete a value from a dictionary, click on delete, then select the item to delete. The item will then be removed from the dictionary.

15.7.7 Schema Information

A compiled EXPRESS schema is treated as a regular design object, but it contains only RoseDomains and RoseAttributes. Schemas may not be modified using the editor.

Each compiled data type is represented by a RoseDomain object. The object contains several different lists of super/sub type information and attribute information.

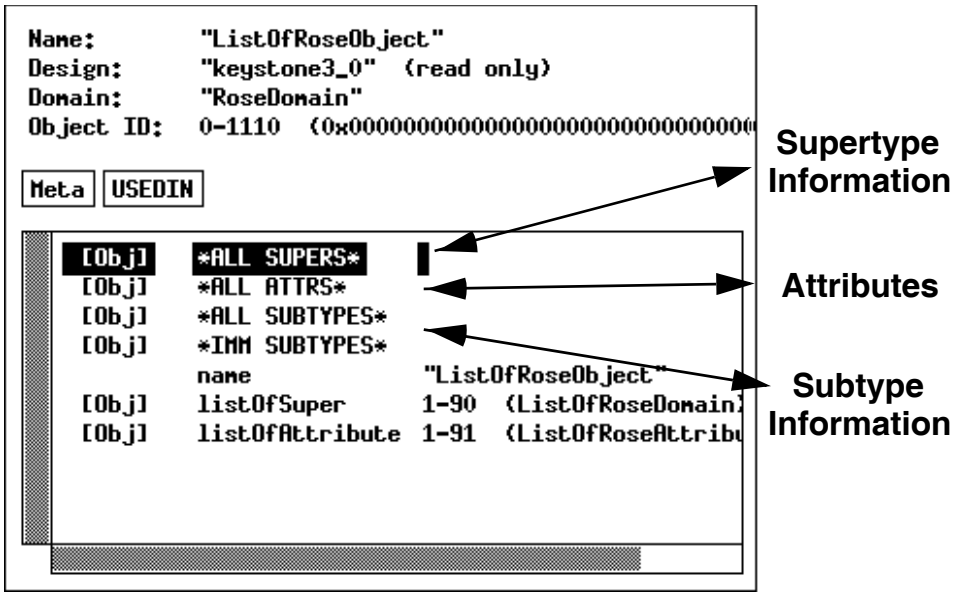


Figure 15.14 — RoseDomain Contents Editor Pane

The display contains:

ALL SUPERS
All Supertypes, from RoseObject down

ALL ATTRS
All Attributes

ALL SUBTYPES
All subtypes

IMM SUBTYPES
The immediate subtypes,

ListOfSuper
The immediate supertypes.

ListOfAttribute
The immediate attributes

16 STEP Part 21 File Browser

16.1 Description

STEP File Browser is a simple utility for viewing STEP Part 21 files under Windows. It uses color to highlight syntactic elements of a file, such as entity IDs, references to entities, and text strings. It also uses entity references as hypertext links to corresponding entity definitions. This allows you to work with the program in a similar way as you would work with a Web browser.

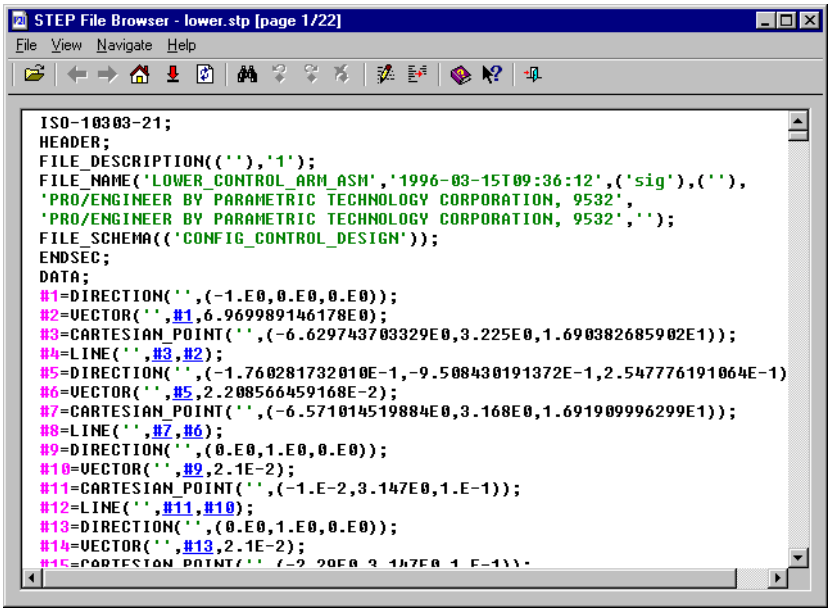


Figure 16.1 — STEP File Browser Window

The program keeps track of your browsing, so you can use the **Navigate | Back** and

Navigate | Forward commands the same way as you do when browsing the Web. It also allows you to use bookmarks to mark lines of the file to quickly return later to the specified position. The bookmarks are sorted so you can use the **Navigate | Previous Bookmark** and the **Navigate | Next Bookmark** commands to move along the sequence of bookmarks.

The browser also allows you to select a subset of entities (apply a working set) to display instead of the whole file. If you select a subset, then all the entities in the subset as well as all the entities they refer to will be displayed in the browser.

You can use any editor you specify (Windows Notepad is the default) to edit the STEP file.

16.2 Command Line

The STEP File Browser is only available on Windows platforms. UNIX users should use the **stepedit** tool instead. The file browser can be started using the following command line:

stepbrws [options] <STEP file name>

The following are the command-line options used by STEP File Browser:

-line <number>
Opens the file and scrolls to the specified line.

-entity <ID>
Opens the file and jumps to the definition of the given entity.

-pagesize <size>
Specifies the size of the file page in kilobytes.

-maxwhole <size>
Specifies the maximum file size in kilobytes to load as a whole.

-help Shows the Help and exit.

16.3 Using STEP File Browser

To open a STEP file choose Open on the File menu or click  on the toolbar. STEP File Browser will show the **Open File** dialog box. Using this dialog box you can locate the file on the disk and open it.

Once the file is open its contents will be displayed with syntax elements highlighted using different colors. You can use the scrollbar, entity references, bookmarks, as well the navigation commands on the **Navigate** menu to move around the file.

Entity references are similar to links on a Web page — when you click them the page will be scrolled to the place where the referred entity is defined. The line with the referred entity will also be marked by a bookmark symbol ♦. This will allow you to return quickly to this place later.

You can also add bookmarks at any line of the file. To add a bookmark move the pointer beyond the left boundary of the text. The pointer will become a  . Click once to add a bookmark at an unmarked line or remove a bookmark from a marked line. Bookmarks are sorted according to their order in the file. Use the **Navigate | Previous Bookmark** and the **Navigate | Next Bookmark** commands to navigate the bookmarks. To remove all bookmarks choose **Remove All Bookmarks** on the **Navigate** menu.

Large files may not be loaded in the browser at once, but in pages of a user-defined size. This allows the browser to use less memory and respond faster to user commands. When you scroll a large file, new pages will be loaded and old ones discarded as soon as you scroll beyond the page boundary. This process is fast enough to be hardly noticeable. The same is true when you use hypertext links or bookmarks.

The page size and the maximum file size before the browser starts using paging can be set using the **View | Options** command or through command-line options. These values are then stored in the system registry, so you only have to specify them when you wish to change them. The values are given in kilobytes. The default page size is 100 kilobytes, and the default maximum file size before the browser begins to split files into pages is 100 kilobytes.

16.4 Menus

16.4.1 File Menu

Open — (Also **<Ctrl+O>**) Shows the **Open File** dialog box. Using this dialog you can browse the file system and locate the file to open.

Reload — (Also **<F5>**) Reloads the open file. This is useful when you used the **Edit** command to edit the file.

Edit — (Also **<Ctrl+E>**) Opens an editor and loads the current file into it. You can specify what editor to use in the **View | Options** command.

Exit — (Also **<Esc>**) Closes the STEP File Browser window and quits the program.

16.4.2 View Menu

Options — (Also **<Alt+Enter>**) Displays the **Options** dialog box (Figure 16.2). Using this dialog box you can select colors for syntax highlighting, default font, paging options, and the editor used to modify STEP files.

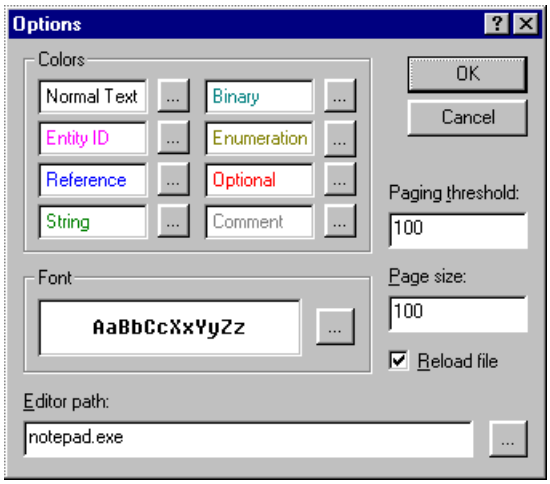


Figure 16.2 — Options Dialog Box

Working Set — (Also **<Ctrl+W>**) Displays the **Working Set** dialog box (Figure 16.3). This dialog box is used to select which entities to include and which to ex-

clude from the working set.

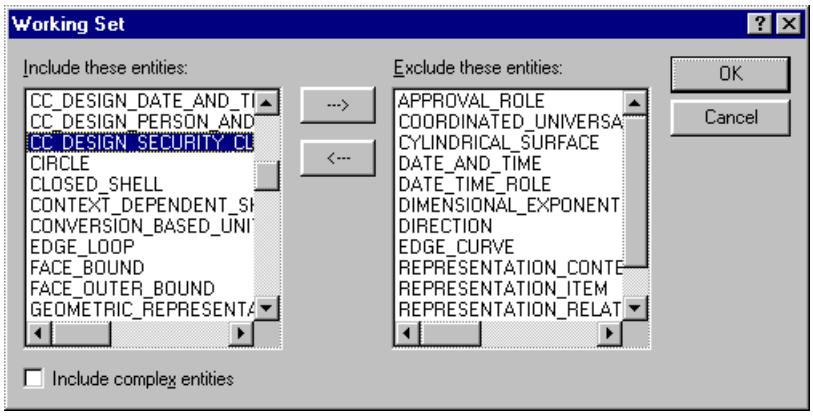


Figure 16.3 — Working Set Dialog Box

Statistics — Displays the **Statistics** dialog box (Figure 16.4). This dialog box contains some statistical information about the file, such as the total number of entities in the file, the number of entity types, and the number of simple and complex entities.

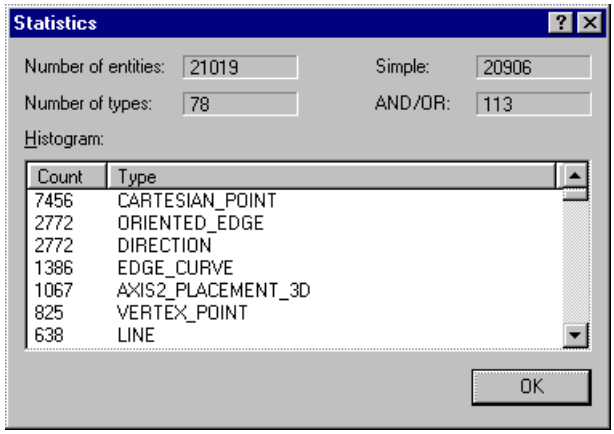


Figure 16.4 — Statistics Dialog Box

16.4.3 Navigate Menu

History — (Also **<Ctrl+H>**) Use this command to display the **History** dialog box with the navigation sequence. A navigation sequence is formed when you move across the file using hypertext jumps at entity references.

The History dialog box contains a navigation sequence list. The list displays the lines from the file at the locations in the navigation sequence. Select an item in the list and then click Go to scroll the view directly to this location. You can also dou-

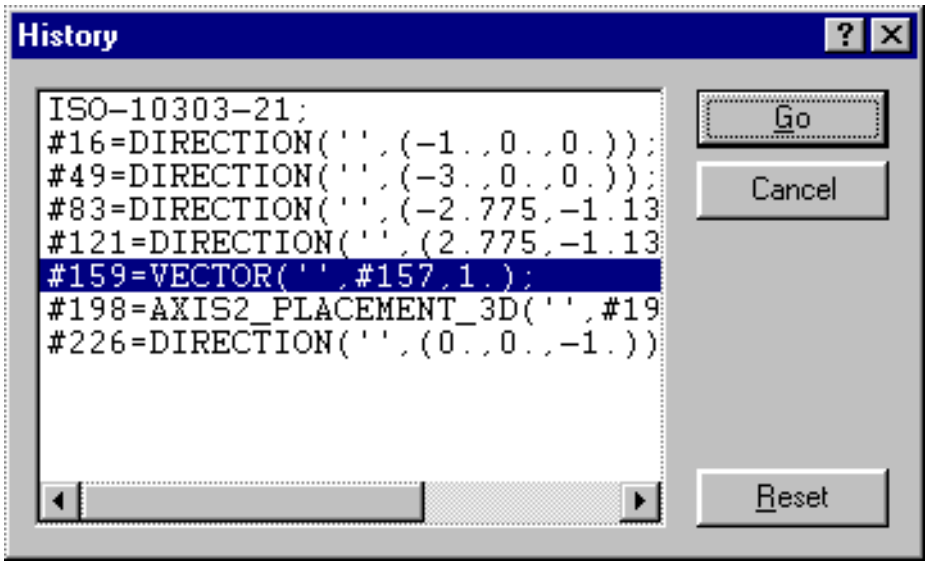


Figure 16.5 — History dialog box

ble-click an item to achieve the same result. Click the Reset button to clear the navigation sequence.

Back — (Also **<Alt+Left Arrow>**) Scrolls to the previous location in the navigation sequence. A navigation sequence is formed when you move across the file using hyperlinks at entity references.

Forward — (Also **<Alt+Right Arrow>**) Scrolls to the next location in the navigation sequence.

Home — (Also **<Home>**) Scrolls to the beginning of the file.

End — (Also **<End>**) Scrolls to the end of the file.

Find — (Also **<Ctrl+F>**) Searches for a specific text in a file. This command displays the **Find** dialog box (Figure 16.6).

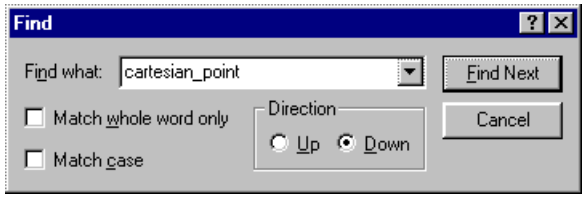


Figure 16.6 — Find Dialog Box

Type the text to find in the **Find what** box. Select the options and the direction to search and click the **Find Next** button.

The **Find what** box keeps a list of the used search patterns. To open the list, click the

button.

Used In — (Also **<Ctrl+U>**) Displays the **Used In** dialog box (Figure 16.7). In the En-

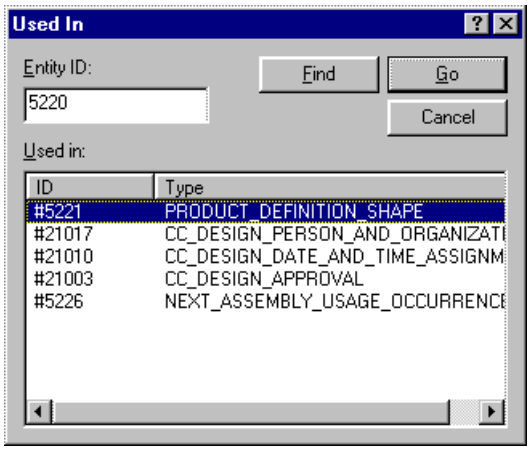


Figure 16.7 — Used In Dialog Box

tity ID box type the entity ID for which you want to find the use list and then click **Find**. The Browser will generate a list of all entities that use the entity you have given. To display one of the found entities, click **Go**.

Previous Bookmark — (Also **<Ctrl+F2>**) Scrolls the text to the location of the previous bookmark.

Next Bookmark — (Also **<Shift+F2>**) Scrolls the text to the location of the next bookmark.

Remove All Bookmarks — Removes all bookmarks from the open file.

16.4.4 Help Menu

STEP File Browser Help — (Also **<F1>**) Shows the context-sensitive Help, which is the help information relevant to the current program context — active window, menu, dialog box, etc.

Contents and Index — (Also **<Ctrl+F1>**) Displays the Help browser. The Help browser is used to navigate through Help topics using table of contents, index, text search, and navigation buttons.

What's This? — (Also **<Shift+F1>**) Puts STEP File Browser into the Help mode.

This command will change the mouse pointer to an ? to indicate that the browser

is in the Help mode. Click a menu command or toolbar button to get information about this command or button.

STEP Tools on the Web — Displays a menu with the following commands:

- **STEP Tools Home Page** — Starts your default browser and connects to the STEP Tools, Inc. homepage.
- **Products** — Starts your default browser and connects to the STEP Tools, Inc. products page.
- **Support** — Starts your default browser and connects to the STEP Tools, Inc. technical support page.

About STEP File Browser — Displays the copyright notice, version number and the license information of your copy of STEP File Browser.

16.5 Context menu

The context menu provides detailed information about a single instance in various forms. Place the cursor over the instance you are interested in, then click the right mouse button to display the context menu.

Used In — This provides a shortcut to the **Used In** dialog box. See Section 16.4.3 on page 149 for more information.

Attributes — Use this command to display the detailed information about entity attributes. This command shows the Entity Attributes dialog box:

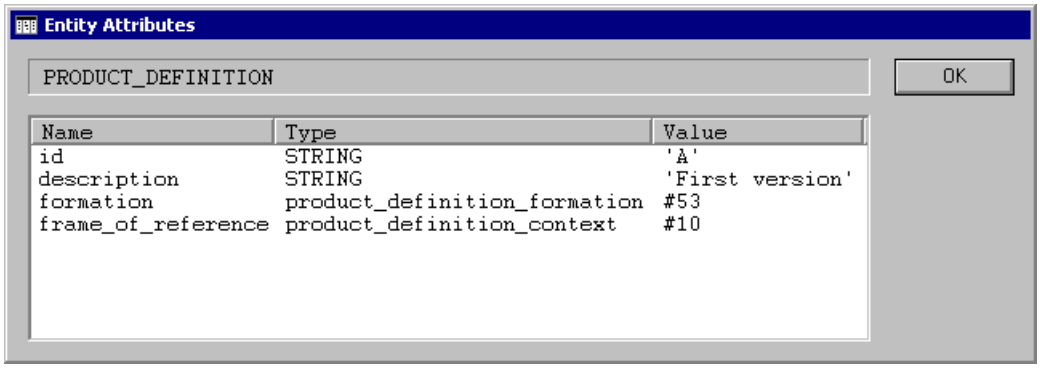


Figure 16.8 — Attributes Dialog Box

This dialog box displays the EXPRESS definitions of entity attributes as described in the file's schema. The name column lists the name of the attributes, the type col-

umn lists the EXPRESS types, and the value column lists their values as defined in the STEP file.

Note: The defined EXPRESS types are not supported, for example for a type defined as:

```
TYPE area_measure = REAL;
END_TYPE;
```

the displayed type will be REAL.

XML — Use this command to view the XML instance data of the selected entity formatted in CEB format. This command shows the XML Instance Data dialog box:

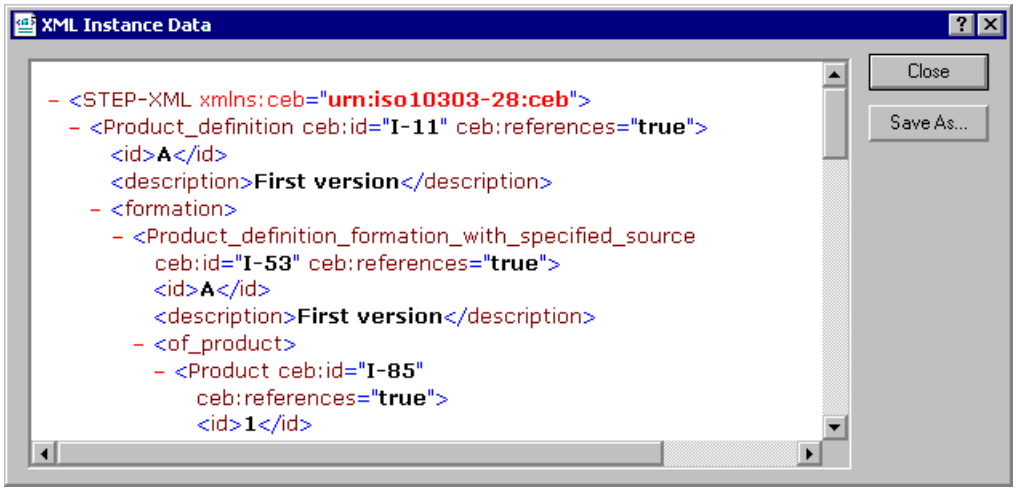


Figure 16.9 — XML Dialog Box

This dialog box displays the XML definition of the entity attributes in the Internet Explorer format. This format allows you to collapse/expand XML nodes by clicking the -/+ signs next to the node names.

16.6 Toolbar

The toolbar is normally placed across the top of the main window, under the menu bar. The toolbar buttons provide quick mouse access to menu commands. The tool-

bar buttons and their corresponding commands are described below.

Click	To
	Open a file.
	View History.
	Go to the previous location in the navigation sequence.
	Go to the next location in the navigation sequence.
	Scroll to the beginning of the file.
	Scroll to the end of the file.
	Reload the file.
	Find a text in the file.
	Scroll to the previous bookmark.
	Scroll to the next bookmark.
	Delete all bookmarks.
	Start the editor to edit the open file.
	Display the **Working Set** dialog box.
	Display the Help browser.
	Put the browser into the Help mode.
	Quit the program.

16.7 Keyboard Shortcuts

The following keyboard shortcuts can be used with STEP File Browser.

Use	To
Alt+Left Arrow	Go to the previous location in the navigation sequence.

Use	To
Alt+Right Arrow	Go to the next location in the navigation sequence.
Up Arrow	Scroll up one line.
Down Arrow	Scroll down one line.
Home	Scroll to the beginning of the file.
End	Scroll to the end of the file.
Page Up	Scroll up one screen.
Page Down	Scroll down one screen.
Ctrl+E	Open the file in an editor.
Ctrl+F	Find a text in the file.
Ctrl+O	Open a STEP file.
Ctrl+U	Find the entities the selected entity is used in.
Ctrl+W	View and edit the working set.
Esc	Quit the browser.
Alt+Enter	Display the **Options** dialog box.
F1	Get help on the current context.
Ctrl+F1	Display the Help browser.
Shift+F1	Put the program into the Help mode.
Ctrl+F2	Go to the previous bookmark.
Shift+F2	Go to the next bookmark.
F5	Reload the open file.

17 STEP Part 28 XML Browser

17.1 Description

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.

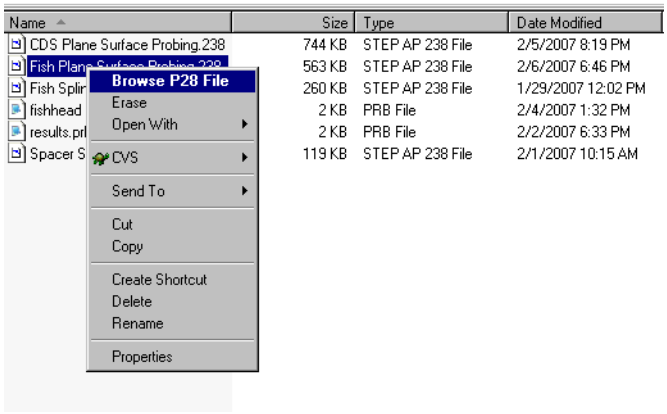


Figure 17.1 — Browse P28 XML

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 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.

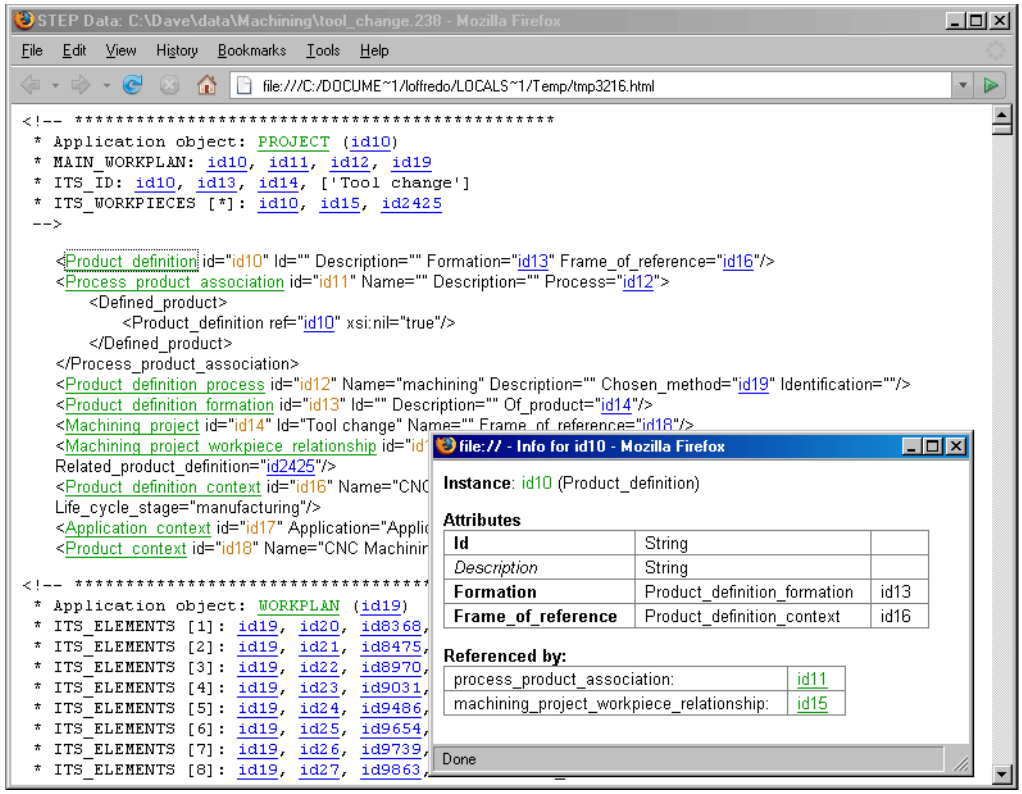


Figure 17.2 — STEP P28 XML In a Web Browser

17.2 Command Line

The STEP Part 28 to HTML conversion utility is available on all platforms. The utility can be run using the following command line:

p28html [options] <Part28-file>

The following are the command-line options used by STEP File Browser:

-help Shows available command line options and exit.

-filter Read from stdin, write to stdout. Other options are ignored.

-o <file> Destination filename for the resulting HTML. By default, the file extension is replaced with ".html" and the result is saved to the current directory. Use a filename of "-" to write to stdout.

17.3 Example

Below are some usage examples for the HTML conversion tool:

```
% p28html sample.p28    ==>  writes sample.html
% p28html sample.xml    ==>  writes sample.html

% p28html -o foo.htm sample.p28    ==>  writes foo.htm
% p28html -o - sample.p28          ==>  writes to stdout

% makep28 | p28html -filter | readhtml ==>  convert in pipeline
```

18 STEP File Cleaner

18.1 Description

A STEP model can contain entity instances which are redundant or irrelevant to the data set being represented. For example, a file may contain duplicate unit definitions, or even entire geometric definitions which are never connected to the product structure in the model. The **stepclean** utility takes such a STEP model and eliminates the superfluous entity instances while keeping the data content of the file intact.

18.2 Command Line

stepclean <data-file> [options]

If no options are selected, the tool will perform all constraint checks. If options are provided, the tool will only perform the requested checks. The options accepted by **stepclean** are:

-help Print this list of options and exit.

-summary give a summary report indicating the total number of instances that were factored, garbage collected, and left for each entity type.

-verbose give detailed messages indicating what instances are removed by the garbage collection and factoring processes.

-config <file>
"use the configuration file specified for the options instead of the default configuration.

-overwrite overwrite the input file, rather than appending a **_clean** suffix to the filename.

-o <file> Specify the output filename. By default, the clean file is saved with the original name, but with **_clean** appended.

-noconfig do not use any configuration file. This option can be used to disable the default configuration that is normally used, if no configuration is provided on the command line.

-factor type [type ...]
specify one or more entity types which are merged when they have the same attribute values

-nofactor type ...
specify one or more entity types which are not merged.

-ignore type.attribute ...
specify an attribute that is ignored when comparing two instances. Two instances are still considered equal even if the values of this attribute differ.

-factor-all factor all entity types not overridden by the **-nofactor** option

-factor-none
factor no entity types except those overridden by the by the **-factor** option

-gcroot type ...
specify a root type from which garbage collection begin

-gcinverse entity.attribute ...
specify an attribute which is traversed in reverse when determining reachability for garbage collection purposes.

18.3 Windows Control Panel

The STEP File Cleaner Windows control panel is shown in Figure 18.1. Run this by selecting **STEP File Cleaner** in the ST-Developer Launcher or the "ST-Developer Tools" Start Menu folder. The following sections describe the fields and setting on

this control panel and show how to perform various tasks with the checker.

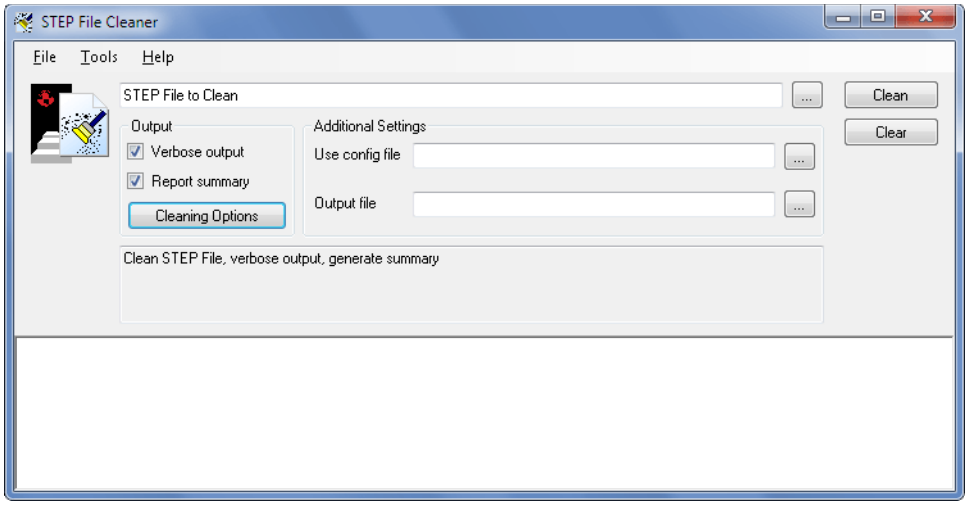


Figure 18.1 — STEP File Cleaner Control Panel

At the top of the control panel is a text field for the STEP files that you would like to check. You can open the file dialog using **Ctrl+O** or the **[...]** button to the right. You can also drag and drop files from the Windows Explorer. Click the **Clean** button to start checking.

18.4 Factoring and Garbage Collection

The **stepclean** utility searches the model for entity instances that have the same attributes values. When it finds two or more instances with the same values, it deletes all but one of the instances, and replaces any references to the deleted instances to the other instance. This process is called factoring. You can specify the types to be included or excluded from the factoring operations. You can also specify attributes which the tool ignores when comparing two instances.

After the **stepclean** tool factors the model, it then performs an optional garbage collection pass over the model. During this process, it deletes any instance that is not accessible from a set of root objects. An object is accessible if it is referenced by a root or another accessible object, or if it references a root or another accessible object through an attribute that has been declared as inverse for the garbage collector.

You must declare one or more root types for the garbage collector to function. If you run **stepclean** without specifying at least one root type either in the control file, or on the command line, the garbage collection operation is not performed.

18.5 Configuration File

The behavior of the cleaner can be controlled using the **Options** dialog, command-line flags, or by collecting the options into a configuratioin file. The **Use config file** field or **-config** command line option give the name of the file.

If no configuration file is specified on the command line, the tool will look for one in **lib/stepclean/{schema}.stc** under the ST-Developer installation (replace **{schema}** with the name of the schema). This behavior can be overridden with the **-no-config** option. Thus, the configuration file for AP203 can be found under the ST-Developer installataion in **lib/stepclean/config_control_design.stc**.

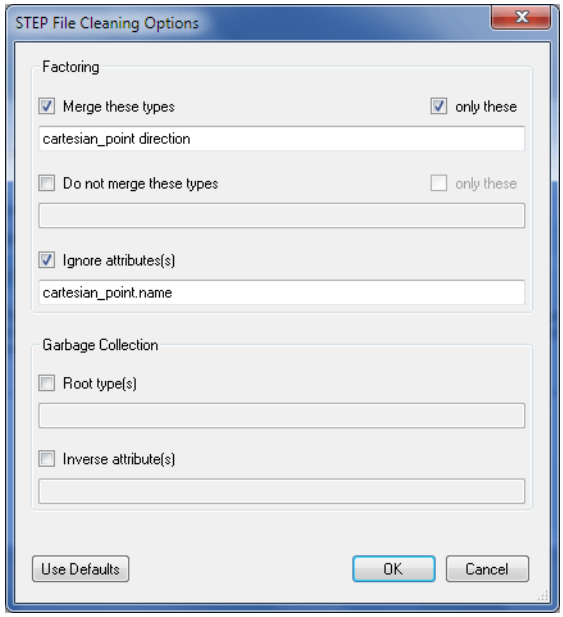


Figure 18.2 — STEP File Cleaner Options

The configuration file is a text file. Comments are introduced with a # character, and continue to the end of the line. Each line of the control file consists of a keyword followed by a set of parameters.

The following is an example of a configuration file:

```
#  Specify the entity types to factor
factor date_and_time person_and_organization organization person

# Set up garbage collection
gcroot product
gcinverse alternate_product_relationship.alternate
alternate_product_relationship.base
gcinverse PRODUCT_RELATED_PRODUCT_CATEGORY.products
gcinverse CC_DESIGN_PERSON_AND_ORGANIZATION_ASSIGNMENT.items
```

```
gcinverse PRODUCT_DEFINITION_FORMATION.of_product
gcinverse CC_DESIGN_APPROVAL.items
gcinverse CC_DESIGN_SECURITY_CLASSIFICATION.items
gcinverse CC_DESIGN_DATE_AND_TIME_ASSIGNMENT.items
gcinverse product_definition.formation
gcinverse property_definition.definition
gcinverse property_definition_representation.definition
gcinverse APPROVAL_PERSON_ORGANIZATION.authorized_approval
gcinverse application_protocol_definition.application
gcinverse approval_date_time.dated_approval
```

The file can contain the following commands:

factor-default
: Set the default to factor entity types that are not otherwise specified. Also controlled by the **-factor-all** or **Do not merge | only these** flags.

nofactor-default
: Set the default to not factor any entities except those specified. Also controlled by the **-factor-none** or **Merge these types | only these** flags.

factor [-exact] type ...
: Specify one or more types to be factored. If the **-exact** flag is given, the type much match exactly, otherwise, the type specified can be a supertype. Also controlled by the **-factor** flag or **Merge these types** field.

nofactor [-exact] type ...
: Specify one or more types not to be factored. If the **-exact** flag is given, the type much match exactly, otherwise, the type specified can be a supertype. Also controlled by the **-nofactor** flag or **Do not merge** field.

ignore ent.att ...
: Specify one or more attributes to be ignored when comparing instances for factoring. Also controlled by the **-ignore** flag or **Ignore attribute(s)** field.

gcroot type ...
: Specify one or more root types for the garbage collector. At least one type must be specified as a root to enable the garbage collector to run. Also controlled by the **-gcroot** flag or **Garbage collection | Root type(s)** field.

gcinverse ent.att ...
: Specify one or more attributes that are inverted for the purposes for garbage collection. When an instance references a live instance via such an attribute, that instance is also live. Also controlled by the **-**

gcinverse flag or **Garbage collection | Inverse attribute(s)** field.

When options conflict, the last option specified in the configaration file takes priority.

Part Four: Data Management and Development Tools

19 C++ Source Management Tools

19.1 Overview

Large information models, such as those in the STEP standard, can contain hundreds of entity definitions. When compiled, these information models will translate into a corresponding number of C++ classes. We have provided several additional tools to make the management of these classes somewhat easier.

- The **extclass** (Section 19.2, pp. 170) and **extall** (Section 19.3, pp. 175) tools place hooks in your generated C++ classes so that you may extend them in a safe manner.

- The **mkmakefile** (Section 19.4, pp. 176) tool will create a makefile for C++ files, such as the classes generated by the EXPRESS compiler. This tool can be used for command line building on UNIX or Windows. On the Windows platform, we have also provided wizards for working with Microsoft Developer Studio.

- The **stepmunch** (Section 19.5, pp. 178) tool can generate calls to the C++ static constructors. This is helpful when linking the C++ classes with code written in other languages, like C, FORTRAN or PL/1. This is not needed on Windows or recent UNIX systems because of improvements in linker/loaders.

19.2 extclass

The **extclass** tool puts **#include** directives into C++ class files generated by the EXPRESS compiler. These include directives are used to extend the classes by placing member variable or member functions into the class declaration.

```
extclass [options] <class>...
```

The options recognized by this utility are:

-help Print usage information and exit.

-d <dir> Look in directory **<dir>** for C++ source and header files to extend. The default is the current directory.

-c / -cx Extend the class definition file **<class>.cxx**. Add an include statement for **<class>.cx** for new function definitions and constructor extensions.

-h This option is equivalent to **-hi -hx**

-hi Extend the class definition file **<class>.h**. Add an include statement for **<class>.hi** for bringing in other **#include** files and ordinary (non-member) function declarations.

-hx Extend the class definition file **<class>.h**. Add an include statement for **<class>.hx** for declaring class member functions and variables.

With no options, **extclass** adds include directives for all extensions.

19.2.1 Description

Code generation is an effective software development technique. Properly used, it reduces development time and maintenance load. However, if the generated code is ever extended, the modifications might be lost if the code is ever regenerated.

The **extclass** tool makes it possible to add extensions to a generated C++ class in such a way that you can regenerate them without losing your changes. The tool adds preprocessor **#include** statements to the class files so that your extensions can be kept in auxiliary files. The auxiliary files are kept under source control and you can regenerate your C++ classes without concern.

19.2.2 Extending the Declaration File (.h)

Declarations for new functions and non-persistent instance variables need be inserted into the class declaration file **<class>.h**. We do this by placing the declarations into auxiliary files. The files we will be working with are:

<class>.h Class declaration file. Generated automatically

<class>.hi Your extensions. Contains new include statements and ordinary function declarations.

<class>.hx Your extensions. Contains new member function declarations and non-persistent instance variables.

As an example, let us examine how we would go about extending a Point C++ class to include some new instance variables and member functions. Given the following EXPRESS definition, we generate a C++ class using the **expfront** tool.

```
ENTITY Point;
        ----------------------------------------
        -- Example Entity declaration:
        ----------------------------------------
        x : REAL;
        y : REAL;
        END_ENTITY;
```

We want to extend the C++ class with some graphics routines from our application program. These routines might require some additional member data and member functions. The first step is to add the hooks into the generated C++ Point class.

```
% cd classes
% extclass Point
```

This adds the hooks to the **Point.h** and **Point.cxx** files. Here we see the Point declaration file. The additions made by the **extclass** tool have been highlighted.

```
#ifndef Point_h
#define Point_h

#include "rose.h"

/* CLASS INCLUDE-FILE EXTENSIONS */
#include "Point.hi"

#define PointOffsets(subClass) \
        RoseStructureOffsets(subClass) \
        ROSE_SUPERCLASS_OFFSET(subClass,Point)

ROSE_DECLARE (Point) : virtual public RoseStructure {
```

```
 private:
        float PERSISTENT_x;
        float PERSISTENT_y;

 public:
        ROSE_DECLARE_MEMBERS(Point);
/* Access and Update Methods */

/* x Access Methods */float x()
{       return ROSE_GET_PRIM (float,PERSISTENT_x);
}
void x (float ax)
{       ROSE_PUT_PRIM (float,PERSISTENT_x,ax); }

/* y Access Methods */
float y()
{       return ROSE_GET_PRIM (float,PERSISTENT_y);
}
void y (float ay)
{       ROSE_PUT_PRIM (float,PERSISTENT_y,ay); }

/* Constructors */
Point ();
Point (float ax, float ay );

/* CLASS DECLARATION EXTENSIONS */
#include "Point.hx"
};
#endif
```

As we can see, the tool has added includes for **Point.hi** and **Point.hx**. The **Point.hi** file will bring in the new include files that we will need for our graphics application.

```
/* File: Point.hi
 *  Application-specific include-files and
 *  ordinary (non-member) function declarations
 *  for the Point class.
 */

#include <graphics.h>/* graphics package decls */
```

The **Point.hx** file will declare the new member functions and instance variables that we need for our extended class. In this example, we add a couple of new fields and a new member function.

```
/* File: Point.hx
 *  Application-specific member functions and
 *  non-persistent instance variables for the
 *  Point class.
 */
BOOL   NP_redraw_state; /* non-persistent data */
Window NP_drawing_window;
```

```
void draw();                /* new member function */
```

Remember that the **Point.hx** file will be included in the middle of the Point class declaration. Any **#include** statements should be placed in the **Point.hi** file and **not** in the **Point.hx** file.

19.2.3 Extending Definition File (.cxx)

Now that we have extended our class declaration, we must finish the job by extending the class definition. The files we will be working with are:

<class>.cxx Class definition file. Generated automatically

<class>.cx Your extensions. Contains new member function definitions and constructor extensions.

In the previous section, we used the extclass tool to add hooks to the **Point.h** and **Point.cxx** files. Here we see the Point definition file. The additions made by the **extclass** tool have been highlighted.

```
#ifndef Point_c
#define Point_c

/* Class Point */
#include "Point.h"

/* CLASS IMPLEMENTATION EXTENSIONS */
#include "Point.cx"

ROSE_BEGIN_MEMBERS ("Point",Point,PointOffsets)
        ROSE_SCHEMA_NAME ("all_types")
        ROSE_VIRTUALSUPERCLASS (RoseStructure)
        ROSE_VARIABLE (float,PERSISTENT_x,"x")
        ROSE_VARIABLE (float,PERSISTENT_y,"y")
ROSE_END_MEMBERS;

/* Default Constructor - This constructor may be modified,
 *  but *DO NOT* add any calls that would initialize ROSE.
 */

#ifndef ROSE_CTOR_EXTENSIONS
#define ROSE_CTOR_EXTENSIONS /* additional initializations */
#endif
Point::Point () {
        PERSISTENT_x = 0;
        PERSISTENT_y = 0;
        ROSE_CTOR_EXTENSIONS;
```

```
}

Point::Point (float ax, float ay )
{
        x (ax);
        y (ay);
        ROSE_CTOR_EXTENSIONS;
}
#endif
```

We will put the member function definitions and constructor extensions in the file **Point.cx**. Since we have added non-persistent instance variables to the class, we must extend the constructors so that these new fields are initialized. We can do this by defining the symbol **ROSE_CTOR_EXTENSIONS** to be the initialization statements for our new variables. The resulting file is as follows:

```
/* File: Point.cx
 *  Member function definitions and constructor
 *  extensions for the Point class.
 */

#define ROSE_CTOR_EXTENSIONS \
              NP_redraw_state = FALSE; \
              NP_drawing_window = NULL;

void Point::draw()
{
       /* C++ code to draw the point on the screen */
}
```

The backslashes in the **#define** statement hide the newlines so that it all looks like one line to the C preprocessor. If you examine the generated code for the constructors, you will see that they each refer to the **ROSE_CTOR_EXTENSIONS** macro. This allows us to easily add new code to the generated constructors without physically editing the files.

19.2.4 Additional Situations

Depending on your extensions, you may not need to use all of the extension files. For example, if all of your new member functions are defined in-line, you may not use a **<class>.cx** file. If your extensions do not require any new header files or typedefs, you may not use a **<class>.hi** file. The **extclass** tool has options to control which hooks are placed in a file. These options are described at the beginning of this section.

Consider the case where your new member functions are all defined in-line. Your extensions may be completely contained in the header files, and so you would not

need a **<class>.cx** file. In this case you would use the **extclass** tool to extend just the header files:

```
% extclass -h <class>
```

This adds include statements for **<class>.hi** and **<class>.hx** to the **<class>.h** file, but leaves the **<class>.cxx** file unchanged.

If your extensions do not require any new header files or typedefs, you might not use a **<class>.hi** file. In this case you would use:

```
% extclass -hx -cx <class>
```

This would add an include statement for **<class>.hx** to the **<class>.h** file, and an include statement for **<class>.cx** to the **<class>.cxx** file.

19.3 extall

The **extall** tool extends all of the class files in a directory, as needed. The tool examines all files in a directory and calls the **extclass** tool as needed when extension files (**.hi**/**.hx**/**.cx**) exist for a class. This tool is quite useful in makefile targets for generating and preparing C++ classes.

extall [options] [<class_dir> ...]

The options recognized by this tool are:

-help Print usage information and exit.

-I<dir> Adds **<dir>** to the search path for extension files. Behaves just like the C or C++ compiler option. By default, the tool just looks in the current directory for extension files.

-d <dir> Look in directory **<dir>** for source files. This is the same as adding the directory to the end of the command line.

The tool goes through each class file in **<class_dir>** and searches for any extension files that match it. If an extension file is found, the **extclass** tool is called to patch in the extension file. By default, the tool searches the current directory for extension files. Additional directories can be specified with the **-I** option. If no directory is specified for class files, the tool looks for a **classes** directory. Given a generated directory of classes (**classes**) and a directory of extension files (**exts**):

```
exts/                   classes/
    foo.hi                  foo.h
    foo.hx                  foo.cxx
```

We could automatically extend all of the class files with:

```
% extall -Iexts  classes
```

This would find the **hi** and **hx** files in **exts** and extend the **foo.h** file in **classes**. There are no **cx** extension files, so **foo.cxx** will not be changed.

19.4 mkmakefile

The **mkmakefile** tool creates a makefile for C++ source files such as the classes generated by the EXPRESS compiler. This is useful for command line builds on either UNIX or Windows. On Windows, the ST-Developer sample projects for Visual Studio offer an alternate mechanism.

mkmakefile [libname]

The options recognized by this tool are:

-help Print usage information and exit.

-o <file> Save makefile to **<file>** rather than **Makefile**/**win32.mak**.

-d <dir> Look in directory **<dir>** for source files, and generate the makefile there as well. Default is the current directory.

-cflags <str> Set the value of the **$(CFLAGS)** macro. All files are given **<str>** as arguments to the compiler.

-includes <str> Set the value of the **$(INCLUDE)** macro. All files are given **<str>** as arguments to the compiler.

-incinst <str> Create an **install-includes** target that copies all generated header files to **<str>**.

-lib <str> Set the library name to **<str>** rather than “Classes”. The name can also be given without any flags as the last argument on the command line.

-libinst <str> Create an **install-library** target that copies the library to **<str>**.

-macos Same as **-unix** below.

-unix Generate a UNIX-style makefile called **Makefile**. This is the default when **mkmakefile** is run on a UNIX box.

-win32 Generate a Windows-style makefile called **win32.mak** suitable for use with NMAKE and other Windows tools. This is the default when **mkmakefile** is run on Windows.

The tool normally generates a makefile appropriate for the platform it is run on, although the **-macos/-unix/-win32** flags can be used to force one or another. On UNIX and MacOS, the generated makefile is called **Makefile**. On Windows, the makefile is called **win32.mak** and can be run using "**nmake /fwin32.mak**".

The generated makefile compiles all C++ source files in a directory and places them into a library. The library will be called "Classes" (**libClasses.a** on UNIX or **Classes.lib** on Windows) unless a different name is given on the command line.

This tool is normally used with the EXPRESS compiler **expfront**. The following example shows how to go from EXPRESS schema to a C++ library:

```
% expfront -classes geometry_schema.exp
% cd classes
% mkmakefile geometry
% make
```

This example generates C++ classes using the EXPRESS compiler and then generates a makefile that will compile all of them. Once this is in place, we call **make** which will compile the classes and collect them into a library called **libgeometry.a**.

The following templates show how to use this within another makefile to generate classes, apply extensions, and build a sub-makefile. Extensions go under source control in a directory called **exts**, so we need to make sure that the compiler can find them. We also construct the makefiles for both platforms so that you can see how to quote everything so that the correct values pass through.

The following is an example of how to do this in a UNIX makefile. Note the use of double quotes in the **-win32** use of ROSE_INCLUDE. This is necessary because of the spaces in directory names.

```
generated_on_unix:
        expfront -writenone -classes schema.exp
        mkmakefile -unix  -d classes \
                -includes '-I../exts -I$$(ROSE_INCLUDE)'
        mkmakefile -win32 -d classes \
                -includes '-I../exts -I"$$(ROSE_INCLUDE)"'
        extall -d classes -Iexts
```

The following is an example of how to do this in a Windows NMAKE makefile.

```
generated_on_windows:
        expfront -writenone -classes schema.exp
        mkmakefile -unix  -d classes \
                -includes "-I../exts -I$$(ROSE_INCLUDE)"
        mkmakefile -win32 -d classes \
                -includes "-I../exts -I\"$$(ROSE_INCLUDE)\""
        extall -d classes -Iexts
```

19.5 stepmunch

stepmunch [function_name] > ctors.c

This utility is useful only in situations when you must link your ROSE C++ application with the something other than the C++ compiler. This might happen when working with code written in other languages, like C, FORTRAN or PL/1. In addition, it is only useful on platforms that do not have "smart" linkers. Windows and some UNIX platforms have linkers that know about initialization functions, and will make sure that they get called regardless of any other factors.

Normally, the C++ compiler puts in code to call the constructors for static objects, but if you are not going to link your application with **CC**, you may need to generate the calls to these static constructors yourself.

The **stepmunch** utility scans the output of the **nm** command for static constructors, and then generates a C function to call them This utility should work with code compiled under either C++ or G++.

The following example scans a library and creates a function called **call_ctors()**, that calls all of the static constructors in the library. Your application should call this function when it first starts up.

```
% nm -g libclasses.a | stepmunch call_ctors > ctors.c
% cc -c ctors.c
```

The file **ctors.o** should then be linked in with your application.

20 ROSE File Tools

20.1 Overview

The ROSE file utility is a command line tool that can perform various services. The following syntax is used to invoke the commands of the ROSE utility:

```
% rose <command> <options>
```

The utility accepts the following commands. A complete description of each can be found in the following sections.

cat	Display the contents of design files.
conflict	Find update conflicts between designs.
create	Create new objects.
delete	Delete objects.
diff	Compare two designs.
format	Change file formats. Can also merge data sets.
help	Print information about tool options.
index	Name objects.
ls	List design files in the search path.
migrate	Migrate data files to a new schema.
mutate	Mutate objects to a new data type.
paths	Display the run-time environment.
reference	Check the validity of inter-file references.
sed	Update a design using output of the diff tool.
set	Set the values of attributes in objects.

20.2 Source Code

The ROSE utility is an application developed using the ROSE Class Library. The source code for a simplified version of this utility is distributed with ST-Developer in the directory **$ROSE/demos/rose**.

We distribute the source code for the ROSE utility to show how you can do similar things with your own STEP applications. Permission to use, copy, modify and distribute the source code in the directory **$ROSE/demos/rose** is granted, provided that the copyright notices appear in all copies and in supporting documentation.

20.3 Using a "Where" Clause

Several of the ROSE file utilities let you select objects that match constraints. The constraints are specified within a "where clause." For example, you could display all Points on the positive Y axis using the following:

```
% rose cat -where "domain='Point' and x=0 and y>0" tutorial1
```

The syntax for the where clause is:

```
<where-clause> = <expr>
<expr> =
    |    <term>
    |    <term> 'AND' <expr>
    |    <term> 'OR' <expr>
<term> =
    |    '(' <expr> ')'
    |    <operand> <rel-op> <operand>
<rel-op> = '<' | '>' | '=' | '<>' | '>=' | '<='
<operand> =
    |    <EXPRESS string>
    |    <integer>
    |    <REAL number>
    |    <attr> <recursive-attr-spec>
<attr> =
    |    'THIS' | 'DOMAIN' | 'DESIGN' | 'SCHEMA'
    |    <an attribute of the object being tested>
<recursive-attr-spec>
    |    '.' <attr> <recursive-attr-spec>
    |    '->' <attr> <recursive-attr-spec>
    |    '[' <integer> ']' <recursive-attr-spec>
    |    '[' <EXPRESS string> ']' <recursive-attr-spec>
    |    nil.
```

20.3.1 Examples

To select all Points whose X coordinate is less than or equal to 0.0:

```
-where "domain='Point' and x<=0.0"
```

To select all instances of objects whose entity definition is in the "topology" schema:

```
-where "this.schema = 'topology'"
```

In this example, the keyword "this" was optional. The same query could have also been written as:

```
-where "schema='topology'"
```

To select all Polygons if the X coordinate of the first vertex is not equal to 0:

```
-where "domain='Polygon' and this.vertices[0].x <>  0.0"
```

To select all objects in which the label is "A little Picture":

```
-where "label='A little Picture'"
```

20.4 rose cat

rose cat [options] <design>...

-where <where-clause>
Specify constraints that must be met. The objects that meet the constraint will be displayed. Refer to **Using a "Where" Clause** (Section 20.3, pp. 180) for additional information.

-v Verbose mode.

This tool to print the contents of a file to standard output. The data is printed in the ROSE text working format. If you specify multiple files, this utility prints each one with separators indicating where one design ends and another begins.

Examples

The following command displays the contents of a STEP data file named

"tutorial1"

```
%rose cat tutorial1

ROSE_OIDS {
    (0 = 0x00807110537000028A2C14F0000622B0400000000)
}
ROSE_DESIGN (RoseDesign
    name: "tutorial1"
    root: $
    keyword_table: $
    name_table: $
    schema : <"picture">
)

STEP_OBJECTS (
    (<0-7> Circle
        radius: 1.5
        center: (<0-1> Point
            x: 2.5
            y: 4))
    (<0-6> Text
        label: "A Little Picture"
        center: (<0-2> Point
            x:5
            y:0))
    (<0-5> Point
        x:0
        y:0)
    (<0-4> Circle
        radius: 1.5
        center: <0-5>)
    (<0-3> Line
        enda: (<0-0> Point
            x: 1
            y: 0)
        endb: <0-1>)
)
```

20.5 rose conflict

rose conflict [-v] <design>|<diffName> <diffName>

-v Print each conflict as it is detected.

The conflict tool finds and displays update conflicts between a design and a diff file, or between two diff files. The types of conflicts that the tool searches for are strictly of a syntactic nature — multiple updates to the same field, updates on a de-

leted object and so forth.

In the example below, two delta files, T1 and T2, each try to change the label and color attributes of a “ColorText” object in the file “original.rose”. The **conflict** tool can be used to detect this.

```
% rose conflict original T1 T2
0 conflicts between original and T1
0 conflicts between original and T2
2 conflicts between T1 and T2
```

Using the **-v** option, you can make the tool display each conflict that is detected.

```
% rose conflict -v backup T1 T2
Modify conflict in:
(<0-6> ColorText
    color: "blue"
    label: "A Little Picture"
    center: <0-2>)

T1                        T2
color="yellow"            color="red"
label="Jack's Picture"    label="Jill's Picture"
```

See Also

rose diff (Section 20.8, pp. 186); **rose sed** (Section 20.17, pp. 193)

20.6 rose create

rose create [options] <schema-name>...

-all Creates one object of every domain in the specified schemas. This is the default behavior.

-domain <domain-name>
Create objects of the specified domain. This overrides the **-all** behavior.

-o <output-name>
Redirects output to **<output-name>**.

-r Recursive mode. For each domain, recursively visits each attribute. If it is an object attribute, this option causes the attribute to be filled

with an object of the appropriate type.

-style <style>
Where **<style>** is either **c++** or **generic_c++**. If this option is specified, the tool generates a C++ program that can be used to create objects.

The **c++** style program uses the C++ class definitions produced by the **express2c++** tool. The **generic_c++** style program uses classes and methods predefined in the ROSE class library.

-v Verbose mode.

Use this tool to create new objects or to generate C++ code for creating new objects. The **-domain** option, if specified, determines the data type of the newly created object.

The argument **<schema-name>** specifies the names of compiled (.rose) schemas to be used for finding data types. You can use the **-all** option to request creation of an object of every type found in the specified schemas.

This tool is primarily used to quickly produce a data set or a C++ program for testing purposes. The generated objects should be structurally correct, but may not meet all of the constraints specified in the EXPRESS schema.

The primitive attribute fields of objects are filled with arbitrary values. If the recursive option (**-r**) is specified, the object attribute fields are filled with an object of the appropriate type; otherwise, the object attribute fields are set to NULL.

Examples

The examples below use the STEP generic resource EXPRESS schema **p42.exp**. These schemas are distributed with ST-Developer in the directory **$ROSE/express/part42**.

To create an instance of every entity defined in the topology schema of Part 42, first compile **p42.exp**, then use the tool as follows:

```
% express2c++ p42.exp
% rose create -all topology_schema.rose
```

The tool can recursively populate the attributes of objects. The following statement will create instances from the geometry schema, because entities in the topology schema use definitions from the geometry schema:

```
% rose create -all -r topology_schema.rose
```

To generate a class based C++ program for creating objects:

```
% rose create -all -style c++ topology_schema.rose
```

20.7 rose delete

rose delete [options] <design>...

-i Interactive mode. Ask whether to delete each object.

-n No execution. Prints objects to be deleted, but do not actually delete them. If the **-v** option is specified, display the objects in ROSE working format; otherwise, just print the OID/name of the object.

-o <output-design>
Place deleted objects into **<output-design>**. The objects are moved to the **<output-design>**.

-where <where-clause>
Constraint that must be met. Delete the objects that meet the constraint. Refer to **Using a "Where" Clause** (Section 20.3, pp. 180) for more on where clauses.

-v Verbose mode.

Use this tool to delete objects from a design. The **-where** option lets you selectively delete objects, and the **-i** option prompts you before deleting each object.

Examples

The following example deletes all Points in the design tutorial1, whose X coordinate is 0.0

```
% rose delete -where "domain=Point AND x=0.0" tutorial1
```

The following example deletes all instances of objects from the topology schema

```
% rose delete -where "domain.schema='topology'" mbb.rose
```

The following example deletes all Lines and Circles.

```
% rose delete -where "domain=Line OR domain=Circle" tutorial1
```

20.8 rose diff

rose diff <design1> <design2> [<output_name>]

-v Verbose mode.

Use the **diff** tool to create a file that describes differences between two designs. This file, when applied to **<design2>** with the sed tool, makes the two designs identical.

If you do not specify an output name, the differences will be stored in the file **diff.rose**.

20.9 rose format

rose format <new-format> [options] <design>...

-o <output-file>
Redirects output to **<output-file>**.

-r Recursive. Merge a data set which is distributed over multiple designs. Specify an output file using the **-o** option.

-v Verbose mode.

This tool to changes the format of a existing design or schema file. This operation reads the file, changes the output format using the **RoseDesign::format()** function, and saves it again. The value of **<new-format>** should be one of the following:

p21 The STEP Part 21 ASCII exchange file format defined in ISO 10303-21. Files are written as **<design>.stp**.

p28 The STEP Part 28 XML file format defined in ISO 10303-28. Files are written as **<design>.p28** and are zip-compressed to reduce size.

p28-raw The STEP Part 28 XML file format as above, but written without any compression. Files are written as **<design>.xml**.

step An alias for the “p21” format described above.

binary ROSE binary working form. Files are written as **<design>.rose**

rose ROSE ASCII working form. Files are written as **<design>.rose**.

express Convert a compiled ROSE data dictionary back into an EXPRESS text description. This option prints to standard output. Only structure is recovered, all constraints and rules are lost.

Examples

The following command converts the file format of the design "tutorial1" to STEP:

```
% rose format step tutorial1
```

Convert the compiled schema "picture" to EXPRESS text:

```
% rose format express -o picture.exp picture
```

To merge the data in files **tutorial1.rose** and **tutorial2.rose**. Place the merged data in the file **merged_data.rose**:

```
% rose format standard -o merged_data tutorial1 tutorial2
```

Data in ROSE working form files can contain inter-file references. Use the **-r** option to recursively visit and merge these references. The example below creates the design file **merged_data2.rose**, which will contain copies of all objects in tutorial1 and all objects that are referenced by tutorial1:

```
% rose format standard -r -o merged_data2 tutorial1
```

20.10 rose help

rose help [<command>]

Use this command to obtain descriptions of the various ROSE file tools. If the parameter **<command>** is not specified, a list of available commands is displayed on the screen.

20.11 rose index

rose index [options] <design>...

-i Interactive. Ask the user to supply a name for each object.

-on [<domain>].attribute
Use the value of the specified attribute as the index name. A domain name can also be specified, to restrict the scope of the attribute.

-where <where-clause>
Constraint that must be met. Names will be assigned only to object that meet the constraint. Refer to **Using a "Where" Clause** (Section 20.3, pp. 180) for more information.

-v Verbose mode.

Use this command to create entries in a design's name table. A design file contains a dictionary that maps names to objects. Named objects can be found quickly and easily within the stepedit tool and within programs written using ST-Developer.

An object can have more than one name. Each name, however, can match only one object. If you assign the same name to many objects, only the last assignment is kept.

Examples

The following creates an index entry for each Person in the employee_db design if the Person's first name is 'Joe':

```
% rose index -on Person.last_name -where "first_name='Joe'" employee_db
```

This uses the value of the "last_name" attribute as the index value. To create an index entry for any object that has a last_name attribute (regardless of the object's type):

```
% rose index -on last_name employee_db
```

This uses the value of the "last_name" attribute as the name of the object. The following finds all Polygons and asks the user to assign a name to each Polygon:

```
% rose index -i -where "domain='Polygon'" my_design
```

20.12 rose ls

rose ls [options]

-l Display files in long format, indicating the owner, date, permissions and size of each database file.

-v Verbose mode.

Use this command to list, by directory, the design files in the search path. The following example shows the result of such a listing:

```
% rose ls

Total number of designs::  4

 In /usr/local/step_toolkit/system_db/schemas
        data_transfer_schema.rose
        ChangeProcessing.rose
        header_section_schema.rose
        express_parser.rose
```

20.13 rose migrate

rose migrate <design>...

-newschema <schema>
Specifies the name of the new schema. This option must be set if the data is in rose or STEP file format.

-v Verbose mode.

Use this command to update a ROSE binary working form file to the most recent version of a schema. To use this effectively, you should understand the impact of modifying data types and the limits of this tool.

Compiled Data Types

The **expfront** tool generates compiled data definitions and places them into compiled ".rose" files. For example, given the following schema, stored in the file **point_schema.exp**:

```
SCHEMA point_schema;
      ENTITY Point;
            x:REAL;
            y:REAL;
      END_ENTITY;
END_SCHEMA;
```

By invoking the EXPRESS compiler,

```
% expfront -rose point_schema
```

you generate a file called **point_schema.rose**. This file contains compiled type definitions. There are two ways to access the definitions in **point_schema.rose**:

- By the entity name (“Point”)
- By the unique object identifier (OID) of the definition object (“#0x008071051700002A7827AF00004CC2040000000010A”)

The STEP Part 21 file format and the ROSE ASCII working form use the first method, while the ROSE binary working form uses the second.

Compiling EXPRESS Schemas

When you recompile an EXPRESS schema, the compiler compares the definitions it had previously generated with the EXPRESS model. If the compiler detects a difference, it regenerates the type definition.

The old and new type definitions are both stored in the compiled schema, allowing applications to use data described by either definition. The new definition is accessible by name or by OID, while the older one is accessible only by OID. Applications and files that refer to types by name will always get the most recent version of the schema.

For example, suppose we added a 'Z' attribute to the Point Entity and recompiled the schema:

```
ENTITY Point;
      x : REAL;
      y : REAL;
      z : REAL;
END_ENTITY;
```

The **point_schema.rose** file will contain two definitions of Point. The “default” definition would be the most recently generated version.

The Migrate Tool

There are three basic formats for storing data:

STEP — the STEP Part 21 physical exchange file format.

This format references data types by name. For example, a Point object might be represented as:

```
#3 = POINT (2.3, 4.0);
```

Attribute values are listed in the order in which they appear in the EXPRESS specification. This means that STEP files are very sensitive to schema modifications.

For example, if we add a Z attribute to our EXPRESS Point definition, we might have trouble reading the above X-Y Point.

ROSE — ROSE ASCII working form

This format references data types by name. For example, a Point object might be represented as:

```
(<1-3> Point x:2.3 y:4.0)
```

The ROSE ASCII working format is more robust than Part 21, because it stores attribute name/value pairs. Adding new attributes, reordering attributes, and deleting attributes will not cause problems. Data in this format will be automatically upgraded to the newest EXPRESS definitions.

For example, we can the above X-Y Point, even after we add a Z attribute to the EXPRESS definition. In this example, the object would be read in as an X-Y-Z Point and a default value would be assigned to the Z attribute.

Standard — ROSE binary working form.

The ROSE binary working form references data definitions by OID. When you modify and recompile a schema, the EXPRESS compiler will generate new type definitions, but will not delete old ones. Thus, binary files can contain data that uses old definitions. The individual objects are not automatically upgraded to the most recent definitions.

For example, suppose we had a data set containing X-Y Point objects. After extending and recompiling our EXPRESS schema to use X-Y-Z Points, We can still read and work with our old X-Y points, but any new Points will be created as X-Y-Z points.

Use the migrate tool to update our old data to the most recent schema definitions. The tool will scan a data set for objects with old definitions and 'migrate' them into the most recent definition. In our example, the tool would find all older X-Y Points, and convert them into X-Y-Z points. The new Z attribute would just contain a default value.

20.14 rose mutate

rose mutate <design> <old_type> <new_type>

-v Verbose mode.

Use this tool to mutate every instance of **<old_type>** in the design **<design>** into an instance of **<new_type>**.

As the name implies, this operation changes the structure of objects. The values of attributes are preserved when possible. The operation is often used to modify objects when a schema is extended.

Examples

The following command reads the design "tutorial1" and converts every instance of "Line" into an instance of "ColorLine"

```
% rose mutate tutorial1 Line ColorLine
```

20.15 rose paths

rose paths

Use this command to display the run-time configuration of your ST-Developer installation. This utility displays the search path that all ST-Developer based applications will utilize by default. (It is possible, of course, for an application to change the path at run time.) This tool is useful for determining from where various schema files are being loaded, especially when you have a compilicated setup.

```
Search Path:          .;.\schemas
System Schema Path:   C:\steptools_9.0\system_db\schemas
Registry Key:         SOFTWARE\STEP Tools, Inc.\ST-Developer\9.0

Complete search path [ROSE.path()]:
        .
        .\schemas
        C:\steptools_8.0\system_db\schemas
```

20.16 rose reference

rose reference [options] <design>...

-oids / -nooids
Check / Do Not Check the OID field of inter-file references. The default is to check (**-oids**).

-r Recursively check referenced design objects.

-v Verbose mode.

Use this command to check the inter-file references of a design. An inter-file (or external) reference is a pointer between an object in one design and an object in another design.

An inter-file reference contains two fields: the name of an external design and the OID of an object within the external design.

This tool can perform several levels of checking. If no options are used, the tool checks the design-name field by finding and loading each externally referenced design. Once an externally referenced design is loaded, the tool checks the validity of the OID field of each inter-file reference.

If the **-nooids** option is specified, the command only verifies that the design of each externally referenced object is in the search path. This checks the design name of inter-file references, but not the OID.

If you use the recursive option (**-r**), whenever one design references another, the referenced design is also checked for dangling external references. In other words, the tool calls itself recursively.

20.17 rose sed

rose sed <design> <diffName> [<outputName>]

-v Verbose mode.

The **sed** tool updates the design using a “diff” file, which is generated by the **diff** tool. The diff file contains objects that describe how to update the objects within the design.

If you do not specify an output name, the original design will be overwritten.

The diff file may contain edit instructions which cannot be applied to the specified design. For example, the diff file may attempt to update an object which has been deleted from the design.

The **sed** tool applies as many edit operations as possible on the specified design. When the tool encounters an operation which cannot be applied, it skips the operation and moves on to the next.

Examples

Suppose we have two engineers working on different copies of the same design. If we want to merge the changes they have made, we can use the **diff** and **sed** tools. The following example computes the changes each engineer has made, and then applies those changes to the original design file. The files T1 and T2 store the changes made by each engineer.

```
% rose diff eng_one original T1
% rose diff eng_two original T2
% cp original.rose backup.rose
% rose sed original T1
% rose sed original T2
```

This example assumes that there will be no conflicts between the two sets of changes. If we expect conflicts, we can use the **rose conflict** tool to detect them.

See Also

rose diff (Section 20.8, pp. 186); **rose conflict** (Section 20.5, pp. 182)

20.18 rose set

rose set <design> <domain> <attr> <value>

-v Verbose mode.

Use the **set** command to change the value of attribute **<attr>** of every object of domain **<domain>** in the design **<design>** to **<value>**.

21 IGES/STEP Converter

21.1 Description

The IGES tools provide a bidirectional translation between the Initial Graphics Exchange Specification and STEP IGES files. The **iges2step** tool reads an IGES file and writes a STEP IGES file. The **step2iges** tool reads a STEP IGES file and writes an IGES file.

21.2 Command Line

iges2step [options] <iges_file>

step2iges [options] <step_file>

Both of these forms are wrappers that invoke the **iges_cvt** executable.

-nonulls With this option, any empty or blank fields within the global or parameter data section of a STEP IGES file are replaced with the appropriate defaults.

-o <output_filename>
Use this option to specify output file name. Output is written with this name and the appropriate extension. Otherwise the input filename is used with the proper extension.

The IGES tools retain only that part of a filename given to it up to, but not including the first '.' It then appends the proper extension when creating the output file.

-v Verbose mode.

The STEP IGES file read and written by the IGES tools has an EXPRESS information model from STEP Tools, Inc. This model captures all of the information described by version 5.1 of IGES. The EXPRESS source file for this information model can be found in **$ROSE/express/local/iges.exp**.

These tools allow ST-Developer applications to access IGES data using the same programming framework as for STEP data. The IGES tools can also be used as the first stage of programs that translate IGES data into the STEP AP Models.

21.3 Windows Control Panel

The IGES/STEP Converter Windows control panel is shown in Figure 21.1. Run this by selecting **IGES/STEP Converter** in the ST-Developer Launcher or the "ST-Developer Tools" Start Menu folder. The following sections describe the fields and setting on this control panel.

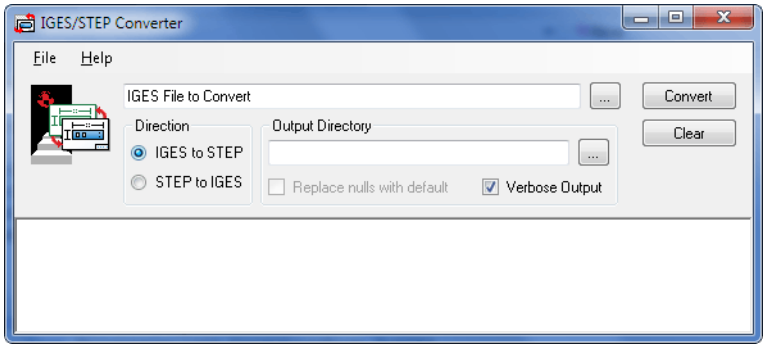


Figure 21.1 — IGES/STEP Converter Control Panel.

At the top of the IGES/STEP Converter control panel is a text field for the STEP or IGES files that you would like to check. You can open the file dialog using **Ctrl+O** or the **[...]** button to the right. You can also drag and drop files from the Windows Explorer. The **Direction** button should change to the appropriate value based on the file extension, but you can change it manually if needed. Click the **Convert** button to start.

The **Output directory** field indicates where the converted file should be placed. If no value is specified, the converted file will be placed in the same directory as the original file.

If **Verbose output** is selected, or the **-v** flag is given, the tool produces a detailed report of its operation.

When converting from STEP to IGES, the **Replace nulls with defaults** field or **-nonulls** flag replaces any empty or blank fields within the global or parameter data section of the STEP IGES file are replaced with the appropriate defaults.

21.4 Examples

Example One — To mix IGES data with other STEP data you can create an information model that references that IGES data. For example, the following information model uses IGES to describe the geometry of a part.

```
INCLUDE 'iges.exp';
SCHEMA Part;
REFERENCE FROM IGES
    (IGESFile);

ENTITY Part_Geometry
    name: STRING;
    owner: STRING;
    geometry: IGESFile;
END_ENTITY;

END_SCHEMA;
```

Example Two — An IGES to STEP translator can be implemented by using the EXPRESS compiler to generate classes for the IGES schema in **iges.exp**. The output of the compiler will be a set of classes that can be used to read and write a STEP IGES file. Semantic translations to the STEP APs may then be implemented by adding methods to the classes. If the result is put into a program called **iges2ap203** then a complete translation form IGES to AP203 can be implemented by typing the following command lines:

```
% iges2step -o a_step_file an_iges_file

# Following is not a STEP Tools, Inc product
% iges2ap203 -o an_ap203_file a_step_file
```

22 DXF/STEP Converter

22.1 Description

The DXF tools offer a bidirectional translation between Drawing Exchange Format (DXF) files and STEP DXF files. The **dxf2step** tool reads a DXF file and writes a STEP DXF file. The **step2dxf** tool reads a STEP DXF file and writes a DXF file. The converter supports AutoCAD Release 11 ASCII DXF format. For more information on this format, refer to the *AutoCAD Release 11 Reference Manual*, Appendix C.

22.2 Command Line

dxf2step [options] filename

step2dxf [options] filename

Both of these forms are wrappers that invoke the **dxf_cvt** executable.

-o <output_filename>

If this option is used, the output file is written with this name and the appropriate extension. Otherwise the input filename is used with the proper extension. The tool retains only that part of a filename given to it up to, but not including the first '.' It then appends the proper extension when creating the output file.

-v Verbose mode, each group's code and data will be printed. Since DXF files can be thousands of lines long, this may produce a large amount of data. For a translation from STEP back to DXF, verbose mode reports every entity, table, or header item that is written.

When converting from DXF to STEP, the tool issues a running count of lines translated, warning and error messages, and a final count of the number of header section items, tables, entities, and blocks translated. When converting from STEP to DXF, the tool reports as it begins to write each section of the file and then how many items it wrote in each section. For more detailed reporting in either mode, use the **-v** option.

The translator will translate all groups supported by AutoCAD Release 11. Optional groups which are not present in the DXF file are assigned their default values in the STEP file. These groups are not written when converting back to DXF if they are set to the default.

Extended Entity Data is supported for every entity. If these groups are present, they are placed, in order, into a list attached to the entity in the **extendHook** attribute. The group code and the data are preserved. No attempt is made to decode the data.

Comments (group code 999) are reported but not saved. There is no way to maintain their context in the STEP database.

22.3 Windows Control Panel

The DXF/STEP Converter Windows control panel is shown in Figure 22.1. Run this by selecting **DXF/STEP Converter** in the ST-Developer Launcher or the "ST-Developer Tools" Start Menu folder. The following sections describe the fields and setting on this control panel.

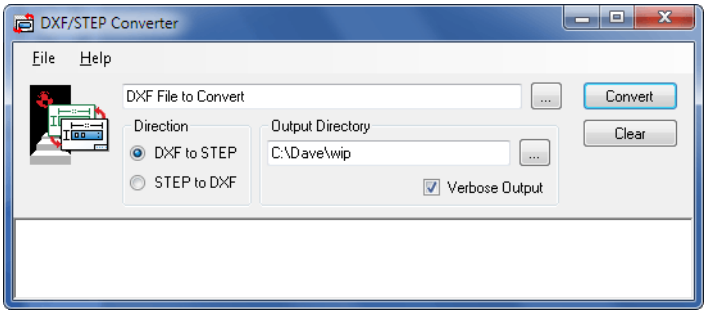


Figure 22.1 — DXF/STEP Converter Control Panel.

At the top of the DXF/STEP Converter control panel is a text field for the STEP or DXF files that you would like to check. You can open the file dialog using **Ctrl+O** or the **[...]** button to the right. You can also drag and drop files from the Windows Explorer. The **Direction** button should change to the appropriate value based on the file extension, but you can change it manually if needed. Click the **Convert** button to start.

The **Output directory** field indicates where the converted file should be placed. If no value is specified, the converted file will be placed in the same directory as the original file.

If **Verbose output** is selected, or the **-v** flag is given, the tool produces a detailed report of its operation.

22.4 Notes

Some assumptions needed to be made during the development of the converter. These assumptions are listed below:

- **VERTEX** and **ATTRIB** entities — In general, these should only follow **POLYLINE** and **INSERT** entities, respectively. However, it isn't specified whether they may stand alone or not. Currently the tool supports them as free-standing entities, if they should occur, by not attaching them to a **POLYLINE** or **INSERT** entity.

- Undocumented Group Codes — The groups for the **DIMENSION** table numbered 145 and up are undocumented (ref. chart on page 530 of *AutoCAD Release 11 Reference Manual*). Currently they represent the types that the codes 100 less represent.

- Optional Groups — Those groups listed as optional are either assigned their default values, or, in the case of a pointer, left pointing to NULL, unless explicitly set by the input file. For Mode 2 translations, the groups are not written if they are set to their default or NULL. This can result in output files being a different size than those originally read in, i.e. if the original CAD program chose to include optional groups which were default.

- Precision — Floating point values are now set to write to 8 places. Occasional precision errors occur in the conversion process, as the input usually contains varying field widths. The errors are primarily in the eighth decimal place.

Index

X