U.S. DEPARTMENT OF COMMERCE
National Institute of Standards and Technology

NISTIR 4351





National PDES Testbed
Report Series

NATIONAL
PDES
TESTBED

# The NIST Working Form for STEP

Revised November, 1990



NIST

U.S. DEPARTMENT OF COMMERCE
National Institute of Standards and Technology

NISTIR 4351

National PDES Testbed

NATIONAL PDES TESTBED

# The NIST Working Form for STEP

Stephen Nowland Clark

U.S. DEPARTMENT OF COMMERCE
Robert A. Mosbacher,
Secretary of Commerce

National Institute of Standards and Technology
John W. Lyons, Director

November 19, 1990



NIST

## Disclaimer

No approval or endorsement of any commercial product by the National Institute of Standards and Technology is intended or implied

Smalltalk-80 is a trademark of ParcPlace Systems, Inc.

Sun-3 and Sun-4 are trademarks of Sun Microsystems, Inc.

Unix is a trademark of AT&T Technologies, Inc.

# The NIST Working Form for STEP

**Stephen Nowland Clark**

# 1 Introduction

STEPparse, the NIST STEP physical file parser, and the associated STEP Working Form, are Public Domain tools for manipulating product models stored in the STEP physical file format [Altemueller88]. These tools are a part of the NIST PDES Toolkit [Clark90a], and are geared particularly toward building STEP translators. The STEP Working Form is an in-memory representation for STEP product models. It relies on the NIST Express Working Form [Clark90b] as an in-core data dictionary, which provides a context in which STEP models can be interpreted. The Working Form code and the STEPparse parser itself are both written to be independent of any particular schema: simply plug in some Express language information model [Schenck90], and the code is ready to run.

A primary goal in the development of STEPparse was to provide a clean back-end interface which would allow various output modules to be easily plugged into the basic front-end parser. To accomplish this, the parser builds up a set of data structures (the STEP Working Form) containing all of the information in a STEP source file. It can then dynamically load one or more output modules. Each module walks through the Working Form, extracting relevant subsets of the available data and producing an appropriately formatted output file. Three STEPparse output modules are provided with the NIST PDES Toolkit: one which produces Smalltalk-80™ object instantiations, one which produces a STEP physical file (so the the Working Form can be used to translate *to* as well as *from* STEP), and one which loads an SQL database from the STEP Working Form [Nickerson90]. The former is used by QDES [Clark90d], a prototype STEP model editor written in Smalltalk-80.

## 1.1 Context

The PDES (Product Data Exchange using STEP) activity is the United States' effort in support of the Standard for the Exchange of Product Model Data (STEP), an emerging international standard for the interchange of product data between various vendors' CAD/CAM systems and other manufacturing-related software [Smith88]. A National PDES Testbed has been established at the National Institute of Standards and Technology to provide testing and validation facilities for the emerging standard. The Testbed is funded by the CALS (Computer-aided Acquisition and Logistic Support) program of the Office of the Secretary of Defense. As part of the testing effort, NIST is charged with providing a software toolkit for manipulating PDES data. This NIST PDES Toolkit is an evolving, research-oriented set of software tools. This document is one of a set of reports which describe various aspects of the Toolkit. An overview of the Toolkit is provided in [Clark90a], along with references to the other documents in the set.

# 2 Implementation Environment

STEPparse and the STEP Working Form were developed on Sun Microsystems Sun-3™ and Sun-4™ workstations running the Unix™ operating system. The parser is implemented in Yacc and Lex, the Unix tools for generating parsers and lexical analyzers. The Working Form data structures are implemented in ANSI Standard C [ANSI89].

The grammar for the language is processed by Bison, the Free Software Foundation's[1] implementation of the Yacc parser generator. The lexical analyzer is produced by Flex[2], a fast, public domain implementation of Lex. The C compiler used is GCC, also a product of the Free Software Foundation, although the Working Form code does not specifically depend on any particular compiler.

# 3 Running STEPparse

STEPparse takes several optional command-line arguments:

```
STEPparse [-d <number>]
          [-e <express>]
          [-s <step>]
```

The `-d` option controls the debugging level; the argument can range from 0 (the default) to 10. The Express schema file is specified with `-e`; if no `-e` option is given, the schema is read from standard input. The STEP input file is specified with `-s`; again, standard input is read if there is no `-s` option. At least one of `-e` or `-s` must be specified; STEPparse cannot read both from standard input.

STEPparse can be built in two different ways, resulting in different interaction patterns. For many applications, a single output module is bound into the translator at build time. In this statically linked case, after the STEP source file has been parsed the user is normally prompted for a single file name. This is the name of the file to which STEPparse's output will be written. In the other (dynamically linked) version, no specific output module is loaded at build time. In this case, after parsing its input, the program asks for an output module. If the file named is an appropriate object file, it is loaded and an output file name requested, which is where the output will be written. Another output module is then requested, and this sequence continues until the user enters an empty line as the name of the output module, which signals STEPparse to exit. This dynamic loading facility is only available under BSD4.2 Unix and its derivates.

1. The Free Software Foundation (FSF) of Cambridge, Massachusetts is responsible for the GNU Project, whose ultimate goal is to provide a free implementation of the Unix operating system and environment. These tools are not in the Public Domain: FSF retains ownership and copyright privileges, but grants free distribution rights under certain terms. At this writing, further information is available by electronic mail on the Internet from gnu@prep.ai.mit.edu.
2. Vern Paxson's Fast Lex is usually distributed with GNU software. It is, however, in the Public Domain, and is not an FSF product. Thus, it does not come under the FSF licensing restrictions.

# 4 Design Overview

The STEP Working Form (WF) is designed in an object-oriented fashion, and is intended to mesh cleanly with the NIST Express Working Form. Indeed, the WF currently relies on the structures of the Express Working Form as an in-memory data dictionary. This section discusses the design of the Working Form, describing STEPparse control flow as well as the data abstractions of the WF. More technical detail can be found in [Clark90c].

## 4.1 STEPparse Control Flow

A STEPparse translator consists of two separate passes: parsing and output generation. The first pass builds an instantiated product model from a STEP source file. This model can then be traversed by an output module in the second pass, producing whatever report is desired.

As currently implemented, STEPparse must, in fact, parse an Express schema (with Fed-X) before it can interpret the constructs in a STEP physical file. To do this, STEPparse first invokes Fed-X's first two passes to build a data dictionary, and then proceeds to parse its STEP source file.

## 4.2 Working Form Data Structures

The STEP Working Form consists of two data abstractions. The Instance abstraction represents individual entity instances in a product model, as well as aggregates and unstructured values (integers, booleans, etc.). A more object-oriented design would clearly break these down into several separate subclasses of Instance; implementation considerations have resulted in a single module. The second abstraction represents a complete product model. This basically consists of an ordered collection of Instances and an Express model to give it a context. The Working Form currently does not record header information (as found in STEP physical files), although this would certainly be useful.

### 4.2.1 Instance

As mentioned above, the Instance abstraction is really the union of several other conceptual classes, representing entity instances, simple-typed data (integer values, booleans, etc.), and various kinds of aggregates. Most of the access functions for this abstraction are restricted to act on Instances of certain classes, which indicates very clearly the need for this module to be broken down into its component classes; this has not been done, primarily because of limitations of the implementation language, C.

Certain attributes are common to all Instances. For example, each instance is marked with a Type, which determines the context(s) in which it can be used. The Type also provides an interpretation for the Instance's value. A user data field is provided so that an arbitrary C pointer can be associated with each Instance in an instantiated model. This allows a Working Form model to be linked to the internal data structures of a solid modeler, for example.

Additionally, every Instance has a value field. The type of this field varies widely with the type of the Instance, but there are three primary classes: simple (unstructured) values, aggregates, and entity instances. Examples of Instances with simple values are numbers, strings, and booleans. These instances each have a single, atomic value of the corresponding C type (int, char*, etc.). An aggregate (which may be an array, bag, list, or set) consists of a collection of values, each of which is itself an Instance. These elements can be accessed via indexing, with valid indices ranging between lower and upper bounds specified by the Express model. Note that these bounds are interpreted differently for different classes of aggregates in Express. The bounds directly specify the range of allowable indices for an array, while they limit the size of other aggregate types. Thus, indices for list, bags, and sets range from 0 to the current size of the aggregate, which in turn must fall between the upper and lower bounds. The Express language also specifies type-specific operations for each class of aggregates, such as intersection and union of sets and bags, and list concatenation. These operations are provided by the STEP WF as the preferred mode of interaction with aggregate Instances. An entity instance's value again consists of a collection of Instances. These are accessed by name, using the attribute names from the entity's class.

Finally, an Instance may have a name. Normally, only external (non-embedded) entities will be named; all other Instances will have NULL names. This is due to the usage prescribed by the STEP Physical File format: an embedded entity cannot be referenced outside of the immediate context in which it is defined, and so has no need for a name. An external entity, on the other hand, can be referenced by any other entity in a product model. This reference requires the entity's name as a handle.

### 4.2.2 Product

The Product abstraction ties things together in the STEP Working Form. This module is used to represent a STEP product model as a whole. A Product consists of a collection of (presumably interconnected) Instances and an Express conceptual schema to give these Instances context. This schema serves as a data dictionary for the Product. External entities in a Product can be looked up by name; other Instances can only be retrieved by coming upon them as components of known Instances.

Externally, a Product looks like a somewhat intelligent container object. New Instances can be added to this container, and existing Instances can be retrieved from it by name. Additional functionality can be gained from the attached Express information model.

# 5 Missing Features

Currently, the Working Form does not handle user-defined entities. STEPparse accepts user-defined entities in a source file, and prints a warning message indicating that they cannot be represented in the Working Form.

As mentioned above, file header information from PDES/STEP physical files is not retained in the Working Form, although STEPparse silently accepts file headers.

Aggregates with non-constant expressions as bounds are not handled properly. Such an aggregate's type information accurately reflects the true upper bound, but the STEP WF routines treat the bound as if it were unspecified. Since unbounded aggregates are dynamically sized, this does not cause memory management problems; the only drawback is that the Working Form does not enforce size constraints on such aggregates.

Comments are currently discarded during lexical analysis, and so currently have no chance to be recorded by the parser. There has been some interest in developing a mechanism through which applications which modify STEP physical files can preserve comments found in the input file.

# 6 Conclusion

The combination of the STEP Working Form with an Express Working Form data dictionary provides a flexible mechanism for performing various manipulations of STEP data in a schema-independent manner. Although it remains to be seem how useful this schema-independence will be in higher-level end-user applications (e.g., design editors, configuration management systems, and process planning systems), the present architecture is quite useful for such generic tasks as translation and database loading.

For further information on STEPparse, the STEP Working Form, or other components of the Toolkit, or to obtain a copy of the software, use the attached order form.

# A References

[Altemueller88] Altemueller, J., The STEP File Structure, ISO TC184/SC4/WG1 Document N279, September, 1988

[ANSI89] American National Standards Institute, Programming Language C, Document ANSI X3.159-1989

[Clark90a] Clark, S. N., An Introduction to The NIST PDES Toolkit, NISTIR 4336, National Institute of Standards and Technology, Gaithersburg, MD, May 1990

[Clark90b] Clark, S.N., Fed-X: The NIST Express Translator, NISTIR 4371, National Institute of Standards and Technology, Gaithersburg, MD, August 1990

[Clark90c] Clark, S.N., NIST STEP Working Form Programmer's Reference, NISTIR 4353, National Institute of Standards and Technology, Gaithersburg, MD, June 1990

[Clark90d] Clark, S.N., QDES User's Guide, NISTIR 4361, National Institute of Standards and Technology, Gaithersburg, MD, June 1990

[Nickerson90] Nickerson, D., The NIST SQL Database Loader: STEP Working Form to SQL, NISTIR 4337, National Institute of Standards and Technology, Gaithersburg, MD, May 1990

[Schenck90] Schenck, D., ed., Exchange of Product Model Data - Part 11: The Express Language, ISO TC184/SC4 Document N64, July 1990

[Smith88] Smith, B., and G. Rinaudot, eds., Product Data Exchange Specification First Working Draft, NISTIR 88-4004, National Institute of Standards and Technology, Gaithersburg, MD, December 1988

ORDER and INFORMATION FORM

## MAIL TO:


National Institute of Standards and Technology
Gaithersburg MD., 20899
Metrology Building, Rm-A127
Attn: Secretary National PDES Testbed
(301) 975-3508


NATIONAL PDES TESTBED

## Please send the following documents and/or software:

- ☐ Clark, S.N., An Introduction to The NIST PDES Toolkit
- ☐ Clark, S.N., The NIST PDES Toolkit: Technical Fundamentals
- ☐ Clark, S.N., Fed-X: The NIST Express Translator
- ☐ Clark, S.N., The NIST Working Form for STEP
- ☐ Clark, S.N., NIST Express Working Form Programmer's Reference
- ☐ Clark, S.N., NIST STEP Working Form Programmer's Reference.
- ☐ Clark, S.N., QDES User's Guide
- ☐ Clark, S.N., QDES Administrative Guide
- ☐ Morris, K.C., Translating Express to SQL: A User's Guide
- ☐ Nickerson, D., The NIST SQL Database Loader: STEP Working Form to SQL
- ☐ Strouse, K., McLay, M., The PDES Testbed User Guide

OTHER (PLEASE SPECIFY)

______________________________

______________________________

______________________________

______________________________

These documents and corresponding software will be available from NTIS in the future. When available, the NTIS ordering information will be forthcoming.

NIST

NIST-114A
(REV. 3-89)

U.S. DEPARTMENT OF COMMERCE
NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY

# BIBLIOGRAPHIC DATA SHEET

1. PUBLICATION OR REPORT NUMBER
NISTIR 4351

2. PERFORMING ORGANIZATION REPORT NUMBER

3. PUBLICATION DATE
DECEMBER 1990

4. TITLE AND SUBTITLE

"The NIST Working Form for STEP"

5. AUTHOR(S)

Stephen Nowland Clark

6. PERFORMING ORGANIZATION (IF JOINT OR OTHER THAN NIST, SEE INSTRUCTIONS)

U.S. DEPARTMENT OF COMMERCE
NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY
GAITHERSBURG, MD 20899

7. CONTRACT/GRANT NUMBER

8. TYPE OF REPORT AND PERIOD COVERED

9. SPONSORING ORGANIZATION NAME AND COMPLETE ADDRESS (STREET, CITY, STATE, ZIP)

10. SUPPLEMENTARY NOTES

☐ DOCUMENT DESCRIBES A COMPUTER PROGRAM; SF-185, FIPS SOFTWARE SUMMARY, IS ATTACHED.

11. ABSTRACT (A 200-WORD OR LESS FACTUAL SUMMARY OF MOST SIGNIFICANT INFORMATION. IF DOCUMENT INCLUDES A SIGNIFICANT BIBLIOGRAPHY OR LITERATURE SURVEY, MENTION IT HERE.)

The Product Data Exchange Specification (PDES) is an emerging standard for the exchange of product information among various manufacturing applications. The neutral exchange medium for PDES product models is the STEP physical file format. The National PDES Testbed at NIST has developed software to manipulate and translate STEP models. This software consists of an in-memory working form and an associated physical file parser, STEPparse. The design and capabilities of STEPparse and of the STEP Working Form are discussed.

12. KEY WORDS (6 TO 12 ENTRIES; ALPHABETICAL ORDER; CAPITALIZE ONLY PROPER NAMES; AND SEPARATE KEY WORDS BY SEMICOLONS)

data modeling; product data exchange; PDES; PDES Implementation tools; schema-independent software; STEP; STEP physical file

13. AVAILABILITY

- [x] UNLIMITED
- [ ] FOR OFFICIAL DISTRIBUTION. DO NOT RELEASE TO NATIONAL TECHNICAL INFORMATION SERVICE (NTIS).
- [ ] ORDER FROM SUPERINTENDENT OF DOCUMENTS, U.S. GOVERNMENT PRINTING OFFICE, WASHINGTON, DC 20402.
- [ ] ORDER FROM NATIONAL TECHNICAL INFORMATION SERVICE (NTIS), SPRINGFIELD, VA 22161.

14. NUMBER OF PRINTED PAGES
11

15. PRICE
A02

ELECTRONIC FORM