The NIST EXPRESS Toolkit
–
Using Applications

Don Libes

Abstract

The NIST EXPRESS Toolkit is a software library for building EXPRESS-related tools. The NIST Part 21 Exchange File Toolkit is a software library for building Part 21-related tools. This paper describes how to use applications built with the toolkits. This includes typical applications such as “fedex” and “p21”, which are stand-alone programs that read and report errors in EXPRESS schemas and Part 21 exchange files.

Readers of this document need no knowledge of the internals of the toolkits.

Keywords: compiler; EXPRESS; implementation; National PDES Testbed; PDES; STEP

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 vendors’ CAD/CAM systems and other manufacturing-related software [1]. 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 Computer-aided Acquisition and Logistic Support (CALS) 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 STEP data. The NIST EXPRESS Toolkit and NIST Part 21 Exchange File Toolkit are a part of this. The toolkits are evolving, research-oriented software tools. This document is one of a set of reports [2][3][4][5][6][7][8][9][10][11][12][13] which describe various aspects of the Toolkit.

Introduction

The NIST EXPRESS Toolkit (called the “toolkit” or the “EXPRESS toolkit” from here on) [2] is a software library for building EXPRESS-related tools [14]. The NIST Part 21 Exchange File Toolkit (called the “P21 toolkit” from here on) [3] is a software library for building Part 21-related tools [15]. This paper describes how to use applications built with the toolkits. This includes typical applications such as “fedex” and “p21”, which are stand-alone programs that read and report errors in EXPRESS schemas and Part 21 exchange files. Interactions with the environment are described. Error messages from the toolkits returned to application users are explained.

Shell commands and output are set in `Courier bold`. EXPRESS source is set in Times Roman (as is the text of this paper).

Using an EXPRESS Toolkit Application

The majority of applications based on the NIST EXPRESS toolkit (from now on simply called "the toolkit") share a common "look and feel". For example, they share a basic set of options that are common to all. This section describes how to invoke these applications.

Applications are started from the command line. Applications read one or more files containing schemas and possibly other information. Application produces output of some kind, but the specific form is application dependent. It can be diagnostics, files, or perhaps a simple status. Any of these can be intended for either human consumption or that of another program.

fedex and Similar Applications

As a specific example, the **`fedex`** program reads a schema file. This may in turn cause it to read other schema files. The files are analyzed for syntactic and semantic errors. If there are no errors, this appears as:

```
% fedex schema.exp
No errors in input
```

The **`%`** is the prompt. (Your prompt may be different.) **`schema.exp`** is the name of the file containing the EXPRESS schema to be analyzed. In this case, no errors were detected. An appropriate message is printed and the exit status is set to 0. The exit status is useful when running applications via a script. For interactive purposes, the exit status has no use and may be ignored (as in this example).

If errors or warnings are detected, they are written to the standard error stream. By default, the standard error stream appears at the terminal. For example:

```
% fedex /etc/passwd
 /etc/passwd:1: --ERROR: syntax error before ‘root’ in express
file /etc/passwd
Errors in input
```

In this case, a syntax error was detected. The error has a standard form. It begins with the file name in which the error occurred. Then a number indicating at which line in the file the error was detected. The word **`ERROR`** or **`WARNING`** appears corresponding to a severity of the problem detected. Finally, the remainder of the line is a description of the error in English.

All toolkit applications print diagnostics in this format. This enables automated environments (see page 3) that can automatically display each diagnostic with its corresponding source line. It is therefore not necessary to have the toolkit itself print the offending line. Having the toolkit generate a listing annotated with diagnostics would actually be distracting. Such a listing would present them in the order matching the lines in the original source file, but this has nothing to do with the order of importance. For example, an incorrect USE statement can cause dozens of errors to appear well before the erroneous statement. It is more efficient to draw the reader's attention first to the incorrect USE statement then any other statements that depend on it, no matter which appears first.

Automating display of diagnostics and erroneous lines with emacs

"GNU emacs" (or just emacs from hereon) [16] is an example of an environment which can interpret toolkit diagnostics in a useful way. In particular, once an application has run and produced diagnostics, the user can move the cursor from one diagnostic to the next in one window, while in a second window, the schema is automatically repositioned to the line that the diagnostic describes.

Using this facility is easier than it sounds. It consists of the following steps:

1. Start emacs.
2. Run the application in emacs' compile mode. To do this, press **`<escape>x-compile<return>`**. You will be prompted "**`Compile command: make`**". Delete the "**`make`**" (using the "usual" editing commands) and replace it with the command line to invoke your application, such as: **`fedex schema.exp`**. A new buffer-window will appear for any diagnostics. The mode-line will say "**`Compilation: run`**".
3. When the application is complete, the mode line will change to "**`Compilation: exit`**".
4. Press **`<control-x><backquote>`**. This will bring up the corresponding source in another window. The contents in the source and diagnostic windows will be aligned so that the first error and its matching source line will appear at the top of their respective windows. You can now edit the source file.
5. Repeat the previous step to go to the next error. When no more errors exist, emacs will report "**`No more errors`**".

External schema references

Referencing a schema that is not defined in the file (or included from another file) causes the toolkit to search for a file with the same name as the schema and with a "**`.exp`**" extension in the directories named by the environment variable **`EXPRESS_PATH`**. For example, in the C-shell, one could say:

```
setenv  EXPRESS_PATH  "/pdes/data/part42  /pdes/data/part202  ~/part"
```

This would define three directories in which to search for schema files. Following UNIX conventions, the first two are absolute references within the file system while the "**`~`**" names a directory relative to a particular user's hierarchy of files. If **`EXPRESS_PATH`** is not set, the default path of "**`.`**" (the current directory) is used.

Imagine a schema being processed contains the following statement:

```
USE  FROM  TOOL_SCHEMA;
```

If TOOL_SCHEMA is already defined, either in a file which was read earlier or the current file, the reference can immediately be completed and no external file need be read. Otherwise, the directories named by **`EXPRESS_PATH`** are searched, in order, for a file by the name "**`tool_schema.exp`**". If such a file is found, it is parsed. If TOOL_SCHEMA is found, the previous USE processing is resumed. If the schema is not found, a diagnostic is issued.

In order to facilitate this, we recommend that all schema files have symbolic links to the names of any schemas within them that are likely to be externally referenced from other schema files. The

symlink program scans a schema file and creates symbolic links corresponding to the internal schema names automatically (see **symlink** below).

For example, stable schemas may have symbolic links placed in a directory of stable schema files, while unstable schemas should be referenced from a specific schema directory. Imagine that the directory for stable schemas is **/pdes/schemas/standard** while **/pdes/schemas/dis** is a directory containing schemas that are still evolving. In this case, the appropriate command might be:

```
setenv  EXPRESS_PATH  "/pdes/schemas/dis /pdes/schemas/standard"
```

The symlink Program

symlink is a program that reads a schema file and creates symbolic links to the schema file. Each link name is derived from a schema name found in the file. For example, if a schema file is called foo.exp and contains schema definitions for A and B, **symlink** creates two links **a.exp** and **b.exp** which both point to **foo.exp**. The symbolic links are created in the current directory.

For example, to create the symbolic links in a directory of schema files, you could use the following C-shell script fragment:

```
foreach file (*.exp)
     symlink $file
end
```

To obtain and build **symlink**, see [7].

The **symlink** program does not require the results of toolkit resolution (i.e., semantic analysis), so it is possible to speed up the process by calling it with the **-r** flag (see page 5).

Common Flags

Most toolkit applications share a common look and feel. In particular, they support a common set of flags. These flags are defined as follows, and further described below.

-v	produces a version description
-d	enables internal debugging information
-p	turns on printing when processing certain objects
-w	enable warnings
–i	ignore warnings
–r	parse, but do not resolve

If an unknown flag is provided, a usage message is printed. This message prints out the flags and options that are appropriate for the application. Applications also generally print out a usage message if no flags or arguments at all are provided. (This is the case with **fedex**.)

The v flag – Version

The **-v** flag causes the application to print out a string describing its version. This will include the application version, the version of the EXPRESS specification, and the version of the toolkit. There is no specific format for the versions.

The **`-v`** flag may be used with other flags or by itself. If used by itself, the application exits immeidately after printing the version.

The d flag – Controlling Debugging

The **`-d`** flag causes some miscellaneous amounts of verification to occur and diagnostics to print during execution. (Do not read too much into this – it is an accumulation of ad hoc debugging tools rather than a rigorous design for debugging.) The "**`-d`**" should be followed by an integer from 0 to 9. Currently, the following values have meaning:

0	print help information on **`-d`** values
1	print information internal to scanner operation
4	malloc and related routines abort if errors are detected in arguments or in the heap
5	same as 4, but the heap is thoroughly examined on every call to malloc and friends
6	same as 5, but just during resolution
8	print information internal to parser operation

The p flag – Controlling Printing During Processing

The **`-p`** flag causes the toolkit to indicate when it is processing certain types of objects. The "**`-p`**" should be followed by a string of characters. Each character causes the inclusion of a class of objects. The characters are as follows:

e	entity
p	procedure
r	rule
f	function
t	type
s	schema or file
#	pass number
E	everything (all of the above)

For example, the command "**`application -p rt#`** ...." would have the application print out a message each time it processes rules and types, plus it would print a message as each new pass is begun.

The w and i flags – Controlling Warnings

The toolkit has the capability to warn the user about constructs that are legal but questionable. All available warnings are printed out in the usage message. Two special warnings are always defined:

`all`	enable all warnings
`none`	disable all warnings

To enable a warning, use the -w flag followed by the name of the warning. To ignore a warning, use the **`-i`** flag followed by the name of the warning.

Any number of `-w` and `-i` warnings may be specified on the command line.

The r flag – Skip Resolution

The default behavior of the toolkit is to parse and resolve schemas. Given the `-r` flag, no resolution will be performed. However, the application-specific backend will still be called.

Exit Status

By default, toolkit applications return values to the environment upon completion. This may be tested in non-interactive scripts.

If no error is encountered, a value of 0 is returned. If the application has a problem interpreting the command-line arguments, a value of 2 is returned. A value of 1 is returned in most other situations. The exit status for applications which abort ungracefully (e.g., dump core) is undefined.

P21 Toolkit Applications

P21 applications are built on top of the EXPRESS toolkit along with the P21 toolkit. They have a similar look and feel to those applications already described. All the same flags are supported. The following additional flags are defined:

`-e`	EXPRESS file for DATA section
`-i`	Part 21 file
`-m`	Estimate of maximum number of instances
`-h`	EXPRESS file for HEADER section

The e flag – EXPRESS file for DATA section

The `-e` flag introduces the name of the EXPRESS file that defines entities in the DATA section of the Part 21 file. This flag is mandatory.

The i flag – Part 21 file

The `-i` flag introduces the name of the P21 file that defines the instances. This flag is mandatory.

The m flag – Estimate of maximum number of instances

The `-m` flag is an estimate of the maximum number of top-level instances that will be created. This number is used to size certain tables which permit particularly efficient processing. It is not necessary to be particularly precise with this estimate. By default, the estimated maximum number of instances is 10000.

The h flag – EXPRESS file for HEADER section

The `-h` flag introduces the name of the EXPRESS file that defines entities in the HEADER section of the Part 21 file. This flag should not normally be used since the definitions are fully provided in the Part21 specification itself. By default, this file is **`p21_header.exp`** which is located according to the usual **`EXPRESS_PATH`** rules (see page 3).

The NIST EXPRESS Server

The NIST EXPRESS Server is a mechanism for using Toolkit tools without installing them locally. Schemas and other data files are e-mailed to the server. The server runs the requested applications on the files and returns any diagnostics, also by e-mail.

For example, analysis of schemas is initiated by sending e-mail to express-server@cme.nist.gov. The subject of the mail is ignored. The body should start with the line:

```
analyze schema.exp
```

where schema.exp is the name of the file to be analyzed. The file itself should then follow in the mail message. The reason for the filename is simply to tag diagnostics with the source file in the usual manner. No signatures are permitted in the message.

Once mail is sent and the server has received the schemas, they are run through fedex and any diagnostics are returned via e-mail.

A variety of other applications are available through the server. The applications typically do not provide the full flexibility that they do when running locally, in part due to security and in part due to minimizing the complexity of the server interface. To get more information on the NIST EXPRESS Server, e-mail the message “help” to the same address. The server is also described in [13].

Interpreting Diagnostics from the EXPRESS Toolkit

The diagnostics produced by the toolkit are intended to be self-explanatory. However, occasional confusion is understandable. For example, some concepts are referred to by different people using different names. For this reason, a list of diagnostics is provided here along with brief explanations. The diagnostics are not presented in any particular order.

Some of these errors should not normally be seen by users. For example, some errors indicate problems internal to the application. The descriptions for such errors are generally aimed at the application writer rather than the user since the application writer is in the best position to fix the problem. A handful of errors indicate internal errors within the toolkit. These can often be provoked by a buggy application, as well as true errors in the toolkit.

Errors not listed are generated by other toolkits or application-specific processing. Internals diagnostics (such as enabled by various **`ifdef`**s) are not described here but can be found by examining the source.

Words in italic are placeholders for the object they name. Placeholders such as *name* and *object* are generic, meaning that any string or object may appear.

If relevant, diagnostics are prefaced with the name and line number of the file where they were detected. Errors produced during scanning (tokenizing) can occasionally produce an error message that references the line after the one with the error. This is because in many cases the scanner can only detect the end of a token by finding the beginning of the next one. If the next token begins on the next line, the line count will mistakenly point to the line after the one where the error actually occurred.

There is a large variety to the grammatical style of the error messages. This is in part due to the numerous packages that comprise the software which cannot be modified. But more typically, unification of such messages is not a high priority task for the toolkit developers.

`Redeclaration of` *name*. **`Previous declaration was on line`** *line*.

Two objects with the same name are declared in the same scope.

`Redeclaration of` *name*. **`Previous declaration was on line`** *line* **`in file`** *file*.

Same as previous message, but declarations are in different files.

`A subordinate failed.`

It is occasionally possible to detect an error but nothing descriptive about it, such as if a subroutine fails in an unexpected way. This error is used in such cases. It is understood that it should be preceded by a more explanatory message from the subordinate.

problem, **`expecting`** *token* **`in`** *scope-type scope-name*

Example: syntax error near 'i', expecting TOK_ASSIGNMENT or TOK_SEMICOLON in function ACYCLIC_MAPPED_REPRESENTATION

The parser detected a problem at the indicated token in the schema. In the example above, the parser knows what tokens can legally follow at any point, but some other token was encountered. The problem can usually be fixed by changing the named token or one slightly earlier. Tokens after the named token will have no effect.

During parsing, all strings in the schema are mapped to tokens, an internal representation that permits an efficient encoding of the file for further manipulation. Generally, the token names are easy to map back to the original string. For example, "TOK_SEMICOLON" is the token for ";" and "TOK_END_FUNCTION" is the token for "END_FUNCTION". Some non-obvious tokens and their token names are:

`||`	**`TOK_CONCAT_OP`**
`**`	**`TOK_EXP`**
`|`	**`TOK_SUCH_THAT`**
`<*`	**`TOK_ALL_IN`**
`:=:`	**`TOK_INST_EQUAL`**
`:<>:`	**`TOK_INST_NOT_EQUAL`**

In addition, all of the built-in procedures (e.g., INSERT, REMOVE) and functions (e.g., LENGTH, NVL) are mapped to the tokens TOK_BUILTIN_PROCEDURE and TOK_BUILTIN_FUNCTION, respectively. All identifiers are mapped to the token TOK_IDENTIFIER.

`unknown warning:` *option*

The -w or -i flag was given but an unrecognized warning was provided.

program name: **`out of space`**

Example: fedex: out of space

The toolkit could not obtain space from the operating system. Try killing other processes on the operating system to release space to the toolkit.

`pausing...press ^C now to enter debugger`

A situation was detected in the toolkit for which no code is currently supplied, either because it is an error or the toolkit author did not foresee the possibility occurring. If run under the debugger, the situation can be explored. Please report the problem (and any solution) to the toolkit maintainer.

`Express Language, DIS (N151), August 31, 1992`

This (or some similar) string describes the current version of the toolkit. There is no specified format for this string. The details of this string are likely to change.

`Errors in input`

The application completed with errors.

`No errors in input`

The application completed without errors.

`Aborting due to internal error(s)`

Immediately after printing this message, the system produces a memory image of itself for debugging, and halts. Please report the problem (and any solution) to the toolkit maintainer.

problem **`in`** *scope name*

Example: syntax error in procedure FOO.

A problem was detected during parsing. The problem is further described by *problem*.

`USE/REF of non-existent object (`*object* **`in schema`** *schema***`)`**

A USE or REFERENCE statement named an object that did not exist.

`Tilde expansion for` *username* **`failed in EXPRESS_PATH environment variable`**

A tilde was followed by a username but no such username is known to the system or the definition of it points to a non-existent or otherwise meaningless directory.

`Schema` *schema* **`was not found in its own schema file (`***filename***`)`**

A schema mentioned in a USE or REFERENCE statement was not present in the original file. It was thus interpreted as a filename which contained the schema. The file was found but no such schema was found within.

`Could not read file` *filename***`:`** *system-error-message*

The named file could not be read. The *system-error-message* describes the reason why.

`Could not write file` *filename***`:`** *system-error-message*

The named file could not be written. The *system-error-message* describes the reason why.

`parse (pass 0)`

Parsing has started.

`pass` *pass*

A new pass has started. The passes are labelled as consecutive numbers. For more information on what each pass does, read the design document [5].

`parse:` *filename* **`(schema file)`**

While parsing, a USE or REFERENCE statement referred to a schema stored in the named file.

`pass` *pass*: *schema* **`(schema reference)`**

While parsing, a USE or REFERENCE statement referred to a schema.

`pass` *pass*: **`(found schema`** *schema* **`in model already)`**

While parsing, a USE or REFERENCE statement referred to a schema. The schema was found in the current file or a file already read.

`pass` *pass*: *filename* **`(schema file not found)`**

While parsing, a USE or REFERENCE statement referred to a schema. The schema was not found even after looking through the file system.

`pass` *pass*: *object-name* **`(`***object-type***`)`**

Example: pass 1: E1 (entity)

The named object is about to be processed in a way appropriate to its type and the current pass.

`Integer expression expected`

The integer value of an expression was requested, but some other type was found. Check the type of the expression.

`Opcode unrecognized while trying to resolve expression`

The opcode of the expression did not make sense.

`Attribute` *attribute* **`cannot be referenced from an aggregate`**

An attribute cannot be referenced from an aggregate (set, bag, etc.) without first selecting from which member of the aggregate the attribute is to be used.

`Attribute` *attribute* **`cannot be referenced from a non-entity`**

The left-hand side of an attribute reference (e.g., “X” in the expression “X.Y”) must be an entity.

`Indexing is only permitted on aggregates`

Only aggregates are indexable. For example, it makes no sense to write “FUNC[1]” where FUNC is a function.

`Enumeration type` *type* **`does not contain item`** *item*

An item has been selected from an enumeration type, but the type has no such item.

`Group reference failed to find entity` *entity*

The right-hand side of a group-reference expression specifies an entity that is not referenceable through inheritance. For example, in the expression “X\Y”, “X” is an entity, but “Y” has no relationship to “X” that permits a group reference.

`Group reference of unusual expression` *expression*

The left-hand side of a group reference expression (e.g., “X” in “X\Y”) is not an entity.

`EXPresolved_op_dot: attribute not an attribute? - press ^C now to trap`

`to debugger`

A situation was detected in the toolkit for which no code is currently supplied, either because it is an error or the toolkit author did not foresee the possibility occurring. If run under the debugger, the situation can be explored. Please report the problem (and any solution) to the toolkit maintainer.

program-name: **`illegal option --`** *option*
`usage:` *usage*

The application command-line was incorrect. For example, it used an undefined option, or an option with a mandatory value had no value.

`Could not open include file '`*filename***`'.`**

An include statement named a file that could not be read.

`unmatched close comment`

A close-comment (i.e., “*)”) was found for which there was no matching open-comment.

`unmatched open comment`

An open-comment (i.e., “(*”) was found for which there was no matching close-comment.

`unterminated string literal`

A string literal had no terminating quote (“'”).

`Empty list in` *function*

An internal error has occurred involving a list which was not expected to be empty.

`Reference to undefined object` *object***`.`**

An object was referred to but was not defined, nor was there any context in which to deduce its type.

`Reference to undefined attribute` *attribute***`.`**

An attribute was referred to but not defined.

`Reference to undefined type` *type***`.`**

A type was referred to but not defined.

`Reference to undefined schema` *schema***`.`**

A schema was referred to but not defined.

`Unknown subtype` *entity* **`for entity`** *entity***`.`**

An entity was referred to as a subtype (using the phrase “SUPERTYPE”) but was not defined.

`Unknown supertype` *entity* **`for entity`** *entity***`.`**

An entity was referred to as a supertype (using the phrase “SUBTYPE”) but was not defined.

`Circularity: definition of` *object* **`references itself.`**

An object was defined in terms of an object referenced (using USE or REFERENCE) from another schema that in turn reference the original object.

`Supertype` *entity* **`is not an entity (line`** *line***`).`**

A supertype declaration (using the phrase “SUBTYPE”) named an object that was not an entity. Supertypes must be entities.

`Subtype` *entity* **`resolves to non-entity`** *object* **`on line`** *line***`.`**

A subtype declaration (using the phrase “SUPERTYPE”) named an object that was not an entity. Subtypes must be entities.

`Unknown attribute` *attribute* **`in entity`** *entity***`.`**

An object was used as if it were an attribute (e.g., UNIQUE clause) but the entity has no such attribute.

`Expected a type (or entity) but` *object* **`is`** *type***`.`**

An object was used as if it were a type or entity (such as an attribute type) but the object was of some other type.

`Function call of` *object* **`which is not a function.`**

An object was used as if it were a function, but it was of some other type.

`Function` *function* **`undefined.`**

A function call was made to a function with no definition.

procedure **`is used as a procedure call but is not defined as one (line`** *line***`).`**

An object was used as if it were a procedure, but it was of some other type.

`No such procedure as` *procedure***`.`**

A procedure call was made to a procedure with no definition.

`Call to` *algorithm* **`uses`** *count* **`arguments, but expected`** *count***`.`**

A function or procedure was called with a differing number of arguments than in its definition.

`Query expression source must be an aggregate.`

The source of a query expression (e.g., “QUERY (variable_id <* source | logical_expression)”) was not an aggregate.

`unexpected type in EXPresolve.  Press ^C now to trap to debugger`

A situation was detected in the toolkit for which no code is currently supplied, either because it is an error or the toolkit author did not foresee the possibility occurring. If run under the debugger, the situation can be explored. Please report the problem (and any solution) to the toolkit maintainer.

`non-hex digit (`*hex***`) in encoded string literal`**

Encoded string literals (such as "01ab0fd3") are hex encodings of strings. If you intended to use an unencoded string, use single quotes instead of double quotes.

`number of digits (`*number***`) in encoded string literal is not divisible by 8`**

Encoded string literals (such as "01ab0fd3") are hex encodings of strings. Such strings must always be divisible by 8, as per §7.5.4 of [14]. If you intended to use an unencoded string, use single quotes instead of double quotes.

`identifier (`*identifier*`) cannot start with underscore`

Identifiers must begin with an alphabetic character. The toolkit continues processing as if underscores were legal.

Interpreting Diagnostics from the P21 Toolkit

`Aggregate instance expected for attribute` *attribute*

An aggregate instance was expected according to the entity definition but something else was found.

`Array expected for attribute` *attribute*

An array instance was expected according to the entity definition but something else was found.

`Bag expected for attribute` *attribute*

A bag instance was expected according to the entity definition but something else was found.

`Attempt to insert into full bag`

Too many instances were added to a bag according to the definition of the size of the bag.

`Boolean expected for attribute` *attribute*

A boolean instance was expected according to the entity definition but something else was found.

`Entity expected for attribute` *attribute*

An entity instance was expected according to the entity definition but something else was found.

`External reference expected for attribute` *attribute*

An external instance was expected according to the entity definition but something else was found.

`Inappropriate entity class for attribute` *attribute*

An entity instance was expected according to the entity definition but something else was found.

`Incompatible types in` *function*

A situation was detected in the toolkit for which no code is currently supplied, either because it is an error or the toolkit author did not foresee the possibility occurring. If run under the debugger, the situation can be explored. Please report the problem (and any solution) to the toolkit maintainer.

`Undefined enumeration value for type` *type*

An enumeration item name was found which was not of the expected enumeration type.

`Aggregate index out of range:` *index*

The index was outside the range permitted by the declaration of the aggregate.

`Not enough attributes given for '@`*`entity-name`*`'`

Not enough attributes were supplied to complete the instantiation of the entity. Remember to include attributes through inheritance.

`Integer expected for attribute` *attribute*

An integer instance was expected according to the entity definition but something else was found.

`Internal instance expected for attribute` *attribute*

An internal instance was expected according to the entity definition but something else was found.

`List expected for attribute` *attribute*

A list instance was expected according to the entity definition but something else was found.

`Logical expected for attribute` *attribute*

A logical instance was expected according to the entity definition but something else was found.

`Number expected for attribute` *attribute*

A number instance was expected according to the entity definition but something else was found.

`Duplicate entry in set`

An instance was added to a set which already contained the same instance.

`Set expected for attribute` *attribute*

A set instance was expected according to the entity definition but something else was found.

`Attempt to insert into full set`

Adding an additional element would cause the number of elements to exceed the range permitted by the declaration of the set (or aggregate).

`String expected for attribute` *attribute*

A string instance was expected according to the entity definition but something else was found.

`Too many attributes given for @`*entity-name*

Too many attributes were supplied to complete the instantiation of the entity. Check the rules on internal/external mapping.

`Reference to undefined entity instance` *instance*

No such entity instance has been defined.

`Reference to unknown entity type` *entity*

No such entity is known to the system. Check the schema files.

`Illegal character encountered while reading file (hex value` *value***`)`**

A character in the P21 file that is not permitted. The hex value is provided since the problem is usually due to control characters that have no printable representation.

`Lowercase character ("`*character*`") not allowed in keyword or enumeration name`

Keywords and enumeration names must be in uppercase. This restriction is mandated by Part 21, but not by Part 11.

function **`is not implemented`**

This function is not implemented either because the toolkit developers did not have the time or funding, or could not conceive of a way to implement the function efficiently, or because the function is not well-defined. If you implement it, please send your implementation to the toolkit developer so that it can be shared with others.

`Expected header entity` *entity* **`but found`** *entity*

The instances in the header section of the file did not match up with the entity definitions themselves. See the **`-h`** option (page 6).

`Part 21, DIS (E), October, 15, 1992`

This (or some similar) string describes the current version of the toolkit. There is no specified format for this string. The details of this string are likely to change.

How to Obtain the Toolkit

The toolkit and its documentation may be obtained in a variety of ways. The simplest way is through anonymous ftp via the Internet. In this case, the source is /pub/step/npttools/exptk.tar. Complete documentation on obtaining the toolkit and its documentation is /pub/step/ntpdocs/exptk-obtaining-installing.ps.Z [5].

Alternately, it possible to receive the toolkit by email. To do this, send the following mail to ntpserver@cme.nist.gov:

```
send step/npttools/exptk.tar.Z
send step/nptdocs/exptk-obtaining-installing.ps.Z
```

If you do not understand these instructions or for any other reason cannot successfully use ftp or email, contact:

FASD – National PDES Testbed
National Institute of Technology and Standards
Bldg 220, Rm A-127
Gaithersburg, MD 20899

npt-info@cme.nist.gov
1-301-975-3508

Questions, Problems, and Support

While we are willing to listen to problems, requests for extension, etc., we cannot guarantee any kind of response. Since the system is distributed in source form, you are encouraged to experiment with the system, especially if you have problems with it. While it is often quicker for you to have us diagnose your problems, it is quicker for us to have you diagnose your own problems. This software is a research prototype, intended to spur development of commercial products.

Nonetheless, if you do have questions and/or problems, you may send e-mail to hotline@cme.nist.gov. Please include schemas, version numbers, platform descriptions, and any other information that could be relevant.

Disclaimer

Trade names and company products are mentioned in the text in order to adequately specify experimental procedures and equipment used. In no case does such identification imply recommendation or endorsement by the National Institute of Standards and Technology, nor does it imply that the products are necessarily the best available for the purpose.

References

[1] International Organization for Standardization, *ISO 10303 Industrial Automation Systems and Integration— Product Data Representation and Exchange — Overview and Fundamental Principles*, Draft International Standard, ISO TC184/SC4, 1992.

[2] Libes, Don, “The NIST EXPRESS Toolkit – Introduction and Overview”, National Institute of Standards and Technology, Gaithersburg, MD, to appear.

[3] Libes, Don, “The NIST STEP Part 21 Exchange File Toolkit:An Update”, National Institute of Standards and Technology, Gaithersburg MD, to appear.

[4] Libes, Don, and Fowler, Jim, “The NIST EXPRESS Toolkit – Requirements”, National Institute of Standards and Technology, Gaithersburg, MD, to appear.

[5] Libes, Don, “The NIST EXPRESS Toolkit - Design and Implementation”, *Proceedings of the Seventh Annual ASME Engineering Database Symposium*, San Diego, CA, August 9-11, 1993.

[6] Libes, Don, and Clark, Steve, “The NIST EXPRESS Toolkit – Lessons Learned”, *Proceedings of the 1992 EXPRESS Users’ Group (EUG ‘92) Conference*, Dallas, Texas, October 17-18, 1992.

[7] Libes, Don, “The NIST EXPRESS Toolkit – Obtaining and Installing”, National Institute of Standards and Technology, Gaithersburg, MD, to appear.

[8] Libes, Don, “The NIST EXPRESS Toolkit – Programmer’s Reference”, National Institute of Standards and Technology, Gaithersburg, MD, to appear.

[9] Libes, Don, “The NIST EXPRESS Toolkit – Creating Applications”, National Institute of Standards and Technology, Gaithersburg, MD, to appear.

[10] Libes, Don, “The NIST EXPRESS Toolkit – Updating Existing Applications”, National Institute of Standards and Technology, Gaithersburg, MD, to appear.

[11] Clark, S.N., “The NIST Working Form for STEP”, NISTIR 4351, National Institute of Standards and Technology, Gaithersburg, MD, November 1990

[12] Clark, S.N., “NIST STEP Working Form Programmer’s Reference”, NISTIR 4353, National Institute of Standards and Technology, Gaithersburg, MD, November, 1990.

[13] Libes, Don, “The NIST EXPRESS Server”, National Institute of Standards and Technology, Gaithersburg, MD, to appear.

[14] Spiby, P., ed., “ISO 10303 Industrial Automation Systems – Product Data Representation and Exchange – Part 11: Description Methods: The EXPRESS Language Reference Manual”, ISO DIS 10303-11:1992(E), July 15, 1992.

[15] ISO 10303 Industrial Automation Systems -- Product Data Representation and Exchange -- Part 21: Clear Text Encoding of the Exchange Structure, Committee Draft ISO TC184/SC4, Van Maanen, Jan, ed., March 12, 1991.

[16] Stallman, Richard M., et al, *GNU’s Bulletin*, Free Software Foundation, Inc., Cambridge, MA, June 1992.