The NIST EXPRESS Server
–
Usage & Implementation

Don Libes

Factory Automation Systems Division
National Institute of Standards and Technology
Gaithersburg, MD 20899

Abstract

The National Institute of Standards and Technology (NIST) has built numerous software toolkits and applications for manipulating STEP and EXPRESS data. The NIST EXPRESS Server is a computational facility at NIST, which provides the ability to run toolkit-based applications remotely without installing them locally. EXPRESS Schemas and other data files are e-mailed to the server. The server runs the requested applications on the files and returns any diagnostics or output, also by e-mail. Applications requiring interaction can either be returned via e-mail so that they can be run locally, or run remotely by telnet or rlogin across the Internet.

Access to the EXPRESS Server is available at zero cost to anyone who can send e-mail. No initial registration is required. Use is anonymous by default, however it is possible to use the server as a collaborative testbed in which case results can be immediately shared with other server users. The server is capable of restricting file access to one user or a subset of users. It is also possible to make files publicly available. The server maintains many STEP-related standards and draft standards for public access. Machine-processable standards such as STEP schemas can be incorporated automatically when processing user files.

The server dramatically lowers the traditional start-up cost and manpower required to obtain and install STEP and EXPRESS tools as well as the continuing support costs to upgrade and maintain the software, by leveraging NIST research, software support and installation, and computing facilities. The server enables people to experiment or demonstrate STEP without a significant investment of time and money, allowing them to build experience and make informed decisions about their future needs for STEP.

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

Background

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). STEP is an emerging international standard for the interchange of product data between various vendors' CAD/CAM systems and other manufacturing-related software [1][2][3]. The National PDES Testbed has been established at the National Institute of Standards and Technology (NIST) 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 software for manipulating STEP data. Provided in the form of tools and toolkits for building new tools, the software is research-oriented and evolving. This document is one of a set of reports ([4] - [14]) which describe various aspects of the software.

Introduction

The National Institute of Standards and Technology (NIST) has built numerous software toolkits and applications for manipulating STEP and EXPRESS data. The NIST EXPRESS Server is a computational facility at NIST, which provides the ability to run toolkit-based applications remotely without installing them locally. EXPRESS Schemas and other data files are e-mailed to the server. The server runs the requested applications on the files and returns any diagnostics or output, also by e-mail. Applications requiring interaction can either be returned via e-mail so that they can be run locally, or run remotely by telnet or rlogin across the Internet.

For example, a request to analyze a single schema is initiated by sending e-mail to **express-server@cme.nist.gov**. The message starts with the line:

```
analyze schema.exp
```

where **schema.exp** is the name of the file to be analyzed. (The filename can be omitted. However, the server will then not identify the file in its responses. This can be confusing if you send multiple requests simultaneously.)

The file itself then follows in the e-mail message. When the server receives the analysis request, the schema is analyzed and any diagnostics are returned via e-mail. A variety of other applications are available through the server.

The functions provided through the EXPRESS server represent only a fraction of the capabilities of the underlying tools used by the server. (For example, the software enables the creation of entirely new tools. However, the server provides no way to exercise this ability.) Nonetheless, the server provides a wealth of services to the public.

In return, NIST benefits by getting very fast feedback on new software releases. If bugs are discovered, new software distributions do not have to be created and distributed. Rather, once the server applications are repaired, all users immediately begin using the new versions. An analogous practice occurs with the draft standard schemas that the server uses.

Access to the EXPRESS Server is available at zero cost to anyone who can send e-mail. No initial registration is required. Use is anonymous by default, however it is possible to use the server as a collaborative testbed in which case results can be immediately shared with other server users. The server is capable of restricting file access to one user or a subset of users. It is also possible to make files publicly available. The server maintains many standards and draft standards for public access. Machine-processable standards such as STEP schemas can be incorporated automatically when processing user files.

The server dramatically lowers the traditional start-up cost and manpower required to obtain and install STEP and EXPRESS tools as well as the continuing support costs to upgrade and maintain the software, by leveraging NIST research, software support and installation, and computing facilities. The server enables people to experiment or demonstrate STEP without a significant investment of time and money, allowing them to build experience and make informed decisions about their future needs for STEP.

For More Information

Contact the Factory Automation Systems Division – National PDES Testbed (1-301-975-3386 or **npt-info@cme.nist.gov**) for more information about the software in general, or other NIST projects at the National PDES Testbed.

This software is a research prototype, intended to spur development of commercial products. Most of the system is available in source form and you are encouraged to obtain and experiment with it.

If you have questions and/or problems, you may send e-mail to the following addresses. Please include schemas, version numbers, platform descriptions, and any other information that could be relevant.

EXPRESS Server	**express-server-admin@cme.nist.gov**
Data Probe	**dprobe@cme.nist.gov**
EXPRESS Analysis	**exptk@cme.nist.gov**
Part 21 Analysis	**p21tk@cme.nist.gov**
Annotated Listing Generator	**shtolo@cme.nist.gov**

Note that the versions of application software used by the server will typically be one release ahead of the software officially available for public distribution.

Applications used by the server that are stable may be obtained through an automated source distribution server at the National PDES Testbed project. The server may be accessed via e-mail to **nptserver@cme.nist.gov**. If you are unfamiliar with the server, send the message "**help**" and you will receive an explanation of how to use it.

Typography and other Conventions

In this document, shell commands and output are set in **Courier bold**. EXPRESS source is set in Times Roman as is the rest of the text. Words or phrases being defined and placeholders that must be replaced by actual data are set in *Times Roman italic*. Optional elements are surrounded by brackets, [such as this phrase]. Occasionally fragments are quoted when they are very small or contain punctuation characters that might otherwise cause them to be confused with the surrounding text.

The NIST EXPRESS Server is simply called "*the server*" in the remainder of this paper.

Cost, Privacy, and Security

There is no cost for using the server.

The server records who uses it but will not retain any private data (e.g., user-provided data files) once a transaction is completed. Logging information is intended only to show use and justify the project to the funding sponsor. Users will never be retroactively billed.

A mechanism is provided to allow joint access or to prevent access between multiple users. See Keys – Storing Multiple Files on page 6.

Services Provided

The following services are currently provided. More services may be provided in the future. The services are described in more detail later in this paper.

- Analysis of EXPRESS schemas
  Schemas are analyzed for syntactic and semantic errors. See Public files on page 9.
- Analysis of Part 21 exchange files
  Part 21 exchange files are analyzed for syntactic and semantic errors. They are also checked against accompanying schemas. See Public files on page 9.
- Creation and/or use of Data Probes
  A Data Probe is a schema-specific editor that can be used to browse or edit entity instances. See Building a Data Probe on page 11.
- Conversion of Short Form to Annotated Listing
  Multiple Integrated Resource Model (IR) schemas are merged and a subset is produced corresponding to the definitions used by a particular Application Protocol (AP) schema. See Converting the Short Form to an Annotated Listing on page 10.
- Public repository for standard and draft standard schemas
  Standard, draft standard, or otherwise interesting schemas may be retrieved or used with other services provided by the server. See Public Files on page 15.
- Sharing and collaboration between other parties
  Users may collaborate on schemas or Part 21 information, using the server as a neutral site. See Keys – Storing Multiple Files on page 6.

Contacting the Server

To use this service, send e-mail to **`express-server@cme.nist.gov`** or **`uunet!cme.nist.gov!express-server`** (UUCP). The "**`Subject:`**" line is ignored.

The return address (or **`Reply-to`** field if given) is used to e-mail responses to. E-mail systems typically provide return addresses automatically. If your e-mail system does not provide a return address, or the provided address is known to be incorrect, you may override the default by giving the following command as the first line of the body of your message:

`path` *<path>*

where *<path>* is the correct address from the server to you. For example:

```
path fred%swill.net@hack.party.edu
```

The server usually responds promptly – within seconds if you are on the Internet, and within a day for most other sites (depending on the number of network gateways between you and the server). Occasional electrical or network problems at NIST can cause substantial delays. While the NIST computers run 365 days a year, weekends are a popular time for NIST-wide power repairs that can temporarily force the server to remain unavailable. We have no control over this, and can only provide this general warning that weekend response time can be unreliable.

If you do not receive a response from the server after several days, then your return address may be incorrect. If the server does not return your e-mail, we probably can't either, so call us on the phone (see For More Information on page 3).

Commands

Most of the services provided by the server are initiated by commands in your e-mail message. Each line of the message is read. If a line begins with a command, the remainder of the line is passed as arguments to the command. Depending upon the command, the remainder of the message will be interpreted as data or additional commands. Commands are case sensitive. It is up to the application as to whether data is case-sensitive or not. All case is preserved.

If an error is encountered, the remainder of the message is discarded, and an appropriate message is returned via e-mail.

Commands	For more information see
analyze	Analyzing Files – Overview on page 9
annotate	Converting the Short Form to an Annotated Listing on page 10
deletekey	Keys – Storing Multiple Files on page 6
get	Receiving a File on page 8
help	Commands on page 5
list	Listing Files on page 8
newkey	Keys – Storing Multiple Files on page 6
path	Contacting the Server on page 4
probe	Building a Data Probe on page 11
put	Explicitly Sending a File on page 7

Additional commands may be added in the future. Any unrecognized command will be treated as a help command. The help command sends back a brief description of all commands and how to use them.

Processing Files In-line

Some commands include files in-line. The filename is one of the command arguments, and the remainder of the e-mail message is taken as the contents of the file.

For example, with one argument, the analyze command (see Analyzing Files – Overview on page 9) takes a filename as argument, immediately followed by the contents of the file.

```
analyze empty-schema.exp
SCHEMA FOO;
END_SCHEMA;
```

The file may also be compressed and uuencoded. For example:

```
analyze empty-schema.exp
begin 664 empty-schema.exp
:’YV04X8@*=(D” @C3Y[L4%#$”9$O 0<67 CY
end
```

Commands which process a single file and then immediately send back the results, such as in this example, delete the input file immediately after the command is completed. (Do not send your only copy....)

Some commands also accept a key parameter (see Keys – Storing Multiple Files on page 6). In this case, the in-line file is saved until the key is later destroyed. If no key is supplied, the filename argument is optional as well. If omitted, the server uses the filename “**`noname.exp`**” to refer to the file in any reports sent back. Thus, for example, the earlier example can be simplified even further:

```
analyze
SCHEMA FOO;
END_SCHEMA;
```

Keys – Storing Multiple Files

It is possible to process multiple files together. There are numerous reasons to split large files up or to keep multiple files separate:

- Many network gateways and mailers enforce length restrictions. 50K files are fine, 64K files are risky from some sites, and 100K are unlikely to arrive from anywhere.
- Schemas may naturally be stored in separate files. In this case, it is desirable for any diagnostics to refer to the original filenames.
- Analysis of a Part 21 file requires an EXPRESS file.
- The server maintains standard and draft-standard schemas already. It is unnecessary to transmit these to the sender or append them to other schemas.

To process multiple files, you must send each file in a separate e-mail message. To avoid conflicts with other users, a key is used to distinguish one user’s files from another’s. Thus, you must first get a key. To get a key, send a message with the command:

```
newkey [<password>]
```

A key will be e-mailed back to you. This key can be used with commands that require a key, such as the command for sending a file. (See “Sending a file”.) For example, the command:

```
newkey
```

would generate a response from the server such as:

```
Your new key is: 17
```

An optional password may be used to reduce the likelihood of other people using your keys. For example, if you send the command:

```
newkey foo
```

you might be told:

`Your new key is: 17.foo`

From then on, you would supply **`17.foo`** wherever a key is normally provided. (The format of keys may not actually look like what is used in these examples – suffice it to say, you should use as a key whatever the server tells you to use as a key.)

When done with the key, you should tell the server that it can delete your files and key. Do this by sending a message with the command:

`deletekey` *<key>*

There is no limit on the number of keys you may create (subject to disk space limitations). For housekeeping purposes, keys and files will be deleted automatically one month after their creation. During disk crises, this lifetime may be capriciously shortened. Key creators are sent e-mail upon deletion of their keys.

Keys may be shared by people collaborating on information stored by the server. Various scenarios are possible:

- Two or more developers from different companies may share files.
  To accomplish this, the first developer simply tells the other developers the key and associated password. Each developer, can then send files to the server which can be retrieved by other developers.
- A developer may create a Data Probe that is usable by others, without providing access to the underlying schema files.
  To accomplish this, the developer creates the Data Probe, deletes the schema files, and then announces the key.
- A developer may request public comments on schemas or other STEP data.
  To accomplish this, the developer downloads the files to a key with no password, and then announces the key.

A more fine-grain sharing policy is not available.

Explicitly Sending a File

Some commands allow files to follow them in-line (see Processing Files In-line on page 5). It is also possible to send a file explicitly. No processing occurs in such a case except that the file is stored at the server.

To send a file, send the following command followed immediately by the data.

`put` *<key>* *<filename>*

The data will be stored and associated with the key and filename. For example:

`put 17 junk.exp`

The software does not actually enforce it, but we recommend that files containing EXPRESS schemas end with "**`.exp`**" while Part 21 files end with "**`.p21`**". Filenames can be constructed of any character but "**`/`**". Filenames may not have a leading "**`.`**".

Filenames may be further suffixed with "**`.#`**" where **`#`** is a number. All such files with otherwise matching names are put together in the order implied by the **`#`**. The first number should be **`0`** (zero).

For example, the files:

`foo.exp.0`
`foo.exp.1`
`foo.exp.2`

will be joined together to make a single file known as "**`foo.exp`**". This joining together actually occurs when an application command is received (**`analyze`**, **`probe`**, etc.).

While not demanded by the server, it is often helpful to compress and uuencode files before sending them to the server. In particular, it

- reduces network usage, which can save you time and money, and
- avoids possibility of truncation or translation problems caused by "intelligent" gateways.

If you want to compress and uuencode files, you should compress, uuencode and *then* split (if necessary) the files, rather than splitting them first. This should save you some effort. By splitting afterwards, you will only have to compress/uuencode once, rather than on each split fragment. With sufficient compression, it may turn out that splitting is not even necessary.

The server will automatically detect files that are uuencoded and compressed, and uudecode and uncompress them. The uuencode label will be ignored in favor of the filename supplied on the command line. (Files should be *both* compressed and uuencoded. Doing one without the other does not make sense.)

Do not append signatures to messages as this will confuse the software. Since the usual signature preface ("**`--`**" on a line by itself) is valid EXPRESS, there is no way for the server to automatically detect and skip signatures.

Listing Files

To list files under a given key, send the command:

`list` <*key*>

To list public files, send the command:

`list`

Receiving a File

It is possible to receive any of the public files or any of your own files. To receive a public file, send the command:

`get` <*filename*>

where <*filename*> is the name of the file.

To receive a file corresponding to a key, send the command:

`get` <*key*> <*filename*>

The file will be compressed and uuencoded. To read it, strip any e-mail headers off of it, uudecode and uncompress it (using the utilities by those names).

If the file is sufficiently large, it will be broken into parts and each part will be e-mailed separately. To recreate the file, strip the e-mail headers from each part and join them together. Then uudecode and uncompress as before.

An automated program to strip the headers and uudecode a group of messages can be received by sending the command "**`send unpack.c`**" to **`library@cme.nist.gov`**.

Files that are encoded or fragmented for transmission will always be prefaced with an explanation (similar to this section) describing how to interpret the rest of the messages that follow.

Public files

Commonly referenced files, such as schemas extracted from various standards (STEP APs, IRs, etc.) are pre-stored in a public area. These files will automatically be used if you have schema references to them that are not otherwise provided by your own schemas.

You can list or retrieve the public files in the public area but you cannot create public files. The other commands' descriptions have the precise details on accessing public files.

Analyzing Files – Overview

The server is capable of analyzing EXPRESS Schemas and Part 21 exchange files for a variety of syntactic and semantic errors and questionable constructions. It is possible to analyze files together, for example, to allow inter-file schema references to be resolved. Analysis of a single file is very simple (see Analyzing a Single File on page 9) while analysis of multiple files is more complex, involving preparing the multiple files and then performing the analysis (see Analyzing Multiple Files on page 10).

A database of standard or draft standard schemas is automatically made available by the server. These schema definitions will be used by default, unless overridden in a user-supplied schema.

Analyzing a Single File

If you are sending a single file for analysis, give the following command as the first line of the body of your message:

`analyze` *<filename>*

where *<filename>* is the name of the file. (Currently, only EXPRESS files are analyzed this way, so *<filename>* should end with "**`.exp`**".) The file itself should follow immediately afterward in the body of the message. For example:

```
analyze empty-schema.exp
SCHEMA FOO;
END_SCHEMA;
```

The file may also be compressed and uuencoded. For example:

```
analyze empty-schema.exp
begin 664 empty-schema.exp
:'YV04X8@*=(D" @C3Y[L4%#$"9$O 0<67 CY
end
```

The file will be deleted immediately after analysis is complete. (Don't send your only copy....)

Analysis of single files generally takes on the order of a few seconds, after which the results are e-mailed back. Any delays are due to e-mail, hardware, or network problems.

Errors and warnings are formatted so that they can be automatically read by emacs compile-mode. In this mode, emacs lets you browse through the diagnostics while it automatically loads and positions the appropriate source file for each diagnostic. This is a significant time-saver. (This also explains why a filename should be supplied even when the file immediately follows – the filename is referred to in the diagnostics.)

Analyzing Multiple Files

To start the analysis, send the command:

> **`analyze`** *<key> <EXPRESS filename> [<Part 21 filename>]*

where *<EXPRESS-filename>* (and optionally, *<Part 21 filename>)* is the name of a file previously sent. If the file references schemas not defined in the current file, a search will be made for a file which has a name matching the unresolved schema, but with a "**`.exp`**" extension. For example, if the current schema encounters the statement:

USE ENTITY FROM FOO;

then the file "**`foo.exp`**" will be loaded and analyzed. If **`foo.exp`** is not associated with the key, the file is loaded from the public files. If **`foo.exp`** contains other schemas, they will also be analyzed.

It is also possible to logically insert other files during analysis by use of an INCLUDE statement. INCLUDE statements were, at one time, valid Express. However, they are not currently. It is best to think of them as a preprocessing phase of the implementation that has nothing to do with the language proper.

With that in mind, INCLUDE statements can appear outside a schema or at the top-level of a schema. Included files are not restricted to including schemas, but may include, for example, a set of entities, a rule, etc. For example:

INCLUDE 'schema-file.exp';

The **`analyze`** command takes an optional argument specifying a Part 21 file. The Part 21 file is analyzed against the given Express file.

Part 21 files place additional constraints against Express files. For example, because Part 21 files provide no entity-to-schema association, they carry an implicit assumption that top-level entity names are unique across all referenced schemas. Thus legal Express files may not be valid in the Part 21 environment.

Converting the Short Form to an Annotated Listing

Standard APs require an annotated listing, also known as "*the long form*". An *annotated listing* is composed of a subset of multiple IRs used by a particular AP. The final form is a listing in the context of a single schema with no cross-schema referencing information. The server generates such a listing with the following command:

> **`annotate`** *[<key>] <schema name> <file name>*

<file name> is the name of the file containing the AP short form. If *<key>* is not present, the remaining lines in the message are used to form the contents of the file.

<schema name> names the particular schema within the file which is to be annotated. A schema name must be provided even if there is only one schema in the file.

If no errors are encountered, the generated annotated listing is placed into a schema called "**`annotated_listing`**". The schema is stored in a file by the name **`annotated_listing.exp`** and e-mailed back.

The presentation of the IRs are regenerated from an internal representation. For this reason, the IR schemas may not superficially resemble their original representation, although they are semantically identical. All comments are stripped from the EXPRESS representing the reference models due to limitations in the underlying schema representation used by the EXPRESS toolkit.

Building a Data Probe

A Data Probe is an EXPRESS schema instance editor and browser [15][16][17]. Currently, a Data Probe can read and write STEP Part 21 exchange files. Data Probes are schema-specific. To build a Data Probe, send the command:

`probe` *[<key>] <filename>*

If a key is provided, *<filename>* is the name of a file previously sent, otherwise anything following is used as the contents of the file.

If no key is provided, the Data Probe is immediately sent back to you as an e-mail message. Currently, Data Probes are provided as SunOS 4.1.2 SPARC stripped executables. The minimum size of a probe is 1.2Mb. (See Receiving a File on page 8 for more information on receiving large files.)

If a key is provided, the Data Probe is stored on the server. You can then ask for it to be sent to you, or (assuming you are on the Internet) you can run it remotely on the EXPRESS server and have it display on your own X server.

The Data Probe is saved as a file called "**`probe`**". Thus it can be retrieved just like any file:

`get` *<key>* **`probe`**

where *<key>* is the key with which the Data Probe was created. (See "Receiving a file" for more information.)

To run the Data Probe remotely, provide the EXPRESS server with the permission to write to your X server. Permission is typically granted by issuing the following command locally:

```
xhost tribble.cme.nist.gov
```

Next telnet or rlogin to the Internet host **`tribble.cme.nist.gov`** and log in as "**`express`**". No password is needed. You will be prompted for the name of your X display, your key and password. If you do not have a password, just press return. (The name of your X display is typically your Internet address followed by "**`:0.0`**", such as "**`snark.cme.nist.gov:0.0`**".) The Data Probe will then run remotely (on **`tribble`**) and display locally on your own X display.

Here is an example of the interaction, beginning from the telnet command on the local host. User keystrokes are underlined.

```
xyzzy% telnet tribble
Trying 129.6.32.54 ...
Connected to tribble.
```

```
Escape character is `^]'.

SunOS UNIX (tribble)

login: express
Last login: Thu Apr  8 14:34:59 from OOPS.NCSL.NIST.G
SunOS Release 4.1.2 (TRIBBLE) #2: Wed Sep 2 12:17:13 EDT 1992
Welcome to the NIST Express Application Server.  If you have
problems or comments, email them to express-server-admin@cme.nist.
gov
This server uses the X window system.  For operation, you must
provide the server with permission to write to your local screen.
To do this, execute "xhost tribble.cme.nist.gov" or its equivalent
on your system.

Enter your X DISPLAY (e.g., bart.uunet.uu.net:0.0): xyzzy.cme.nist
.gov:0.0
Enter key (without password).  This will have been previously
given to you from the NIST EXPRESS server.  Enter key: 111
Enter password: ________
```

The password is not echoed. The Data Probe starts at this point.

Expect [18] is a scripting language which enables automation of interactive programs. Here is an Expect script called **autoprobe** which automates the task of logging in. The script takes a key as the first argument and an optional password as the second argument.

```
#!/usr/bin/expect
if {[llength $argv] < 2]} {
      send_error "usage: autoprobe key \[password]\n"   exit
}
set timeout -1
exec xhost tribble.cme.nist.gov
spawn telnet tribble.cme.nist.gov
expect "login: " {send "express\r"}
expect "DISPLAY*: " {
      send "[exec hostname].[exec domainname]:0.0\r"
}
expect "key: " {
      send "[lindex $argv 1]\r"
}
expect "password: "  {
      if {[llength $argv] > 2} {
            send "[lindex $argv 2]\r"
      } else {
            send "\r"
```

```
        }
    }
    expect
```

Implementation Notes

The following sections of the paper describe implementation aspects of the server software itself. These notes are of primary relevance to the server administrator. Currently, the server software is not available for public distribution. Nonetheless, these notes may be useful to others wishing to better understand this server or anyone designing their own server.

Server

The server runs on tribble.cme.nist.gov, a Sun SPARCstation II owned and supported by the Factory Automation Systems Division at NIST. The machine is physically located in the Metrology Building at the NIST campus in Gaithersburg, Maryland. All file references in the remaining sections of this paper are assumed to be from **`tribble.cme.nist.gov`**.

To preserve good response time, users are prevented from logging in when three or more users are effectively using 100% of the CPU time for more than one minute. This figure was not scientifically determined and may be readjusted as we gain experience with the server.

Software

The server is written in approximately 1000 lines of Expect. The code to handle e-mail requests is 900 lines while the code executed at login (to run a Data Probe) is 100 lines. Application-specific processing is performed by toolkit software written in C and C++. These commands include:

`fedex`	EXPRESS schema analyzer
`p21`	Part 21 analyzer
`mkprobe`	Data Probe creation software
`probe`	Data Probe itself (one for each schema)
`shtolo`	Short to annotated listing generator

The executable code for the server is in **`/proj/elib/services/express`**. In this directory are the following files:

`README`	brief description of files in the directory
`common.exp`	common definitions for **`login.exp`** and **`server`**[1]
`login.exp`	script run when a person logs into **`tribble`** as user "**`express`**"
`mkProbe`	symbolic link to real **`mkProbe`**[2]
`p11`	symbolic link to real **`fedex`**
`p21`	symbolic link to real **`p21`** program

1. Both Expect scripts and EXPRESS files are optionally suffixed with a ".exp" extension for readability. Unfortunately, the benefit fails entirely in situations like these.
2. The "real" executables are stored in a common bin directory elsewhere. This was done out of habit. It is standard practice for large software package and may turn out to be unnecessary for the server.

pub	public EXPRESS schemas. For more information, see Public Files on page 15. This directory includes:	
	README	brief description of files in the directory
	p21_header.exp	header schema defined in Part 21
	other EXPRESS files	
relink	script to recreate all the symbolic links in the pub directory	
server	script run when e-mail is received	
server.help	help file for server	
shtolo	symbolic link to real **shtolo** program	
symlink	script to create symbolic links corresponding to all the schema names in a single EXPRESS file	
spool	where user-writable files are kept. This directory includes:	
	key	contains last key used
	log	log of activity. See Log on page 14.
	key-specific directories	See Key-specific Directories on page 15.

Configuration

A number of configuration parameters are stored in the file **common.exp**. These include the location of the spool and public directory and the log file. Also parameterized is the garbage collection frequency, server administrator, format of the log file, and the EXPRESS server host itself.

The server and the login code each customize parameters specific to themselves. For example, the server parameterizes the location of the key file, help file, as well as all the applications.

Error Handling

Server error handling is robust. There are several types of errors that are handled by the server:

- User-Server Errors
  The server reports user-server errors (e.g., unknown command) by e-mail back to the user, with an appropriate explanation of the error.
- User-Application Errors
  The server reports user-application errors (e.g., improper application usage) by e-mail back to the user, with whatever diagnostic is produced by the application.
- Internal Application Errors
  Internal errors in an application are e-mailed back to the user. The user should report the problem to the application maintainer. See For More Information on page 3.
- Internal Server Errors
  Internal diagnostics generated by the server are e-mailed to the server administrator. A brief note is also e-mailed back to the user saying that the server administrator has been alerted to the problem.

Log

Each command (or login) generates an entry in the log. Here are several sample entries:

```
Wed Apr  7 19:28:46 EDT 1993;libes@cme.nist.gov (don libes);m;newkey
Wed Apr  7 19:28:48 EDT 1993;libes@cme.nist.gov (don libes);m;newkey is
                  122
Wed Apr  7 19:34:07 EDT 1993;libes;m;list 111
Thu Apr  8 02:17:58 EDT 1993;"douglas j. martin" <70412.3036@com-
                  puserve.com>;m;path 70412.3036@compuserve.com
Thu Apr  8 02:17:58 EDT 1993;70412.3036@compuserve.com;m;analyze pipe.-
                  exz
Thu Apr  8 02:48:24 EDT 1993;"douglas j. martin" <70412.3036@com-
                  puserve.com>;m;path 70412.3036@compuserve.com
Thu Apr  8 03:01:33 EDT 1993;droid.cme.nist.gov:0.0;l;probe, key 17
```

Each entry is a set of fields delimited by semicolons. The first field is the date when the entry was made. The second is the e-mail address and an optional user name, or the **DISPLAY** value for logins. The next field is the letter **m** if the request came via e-mail or **l** for login. The last field is the actual command or an appropriate comment generated by the server.

The log must be world-writable. It should be cleaned out occasionally since it will grow without bounds.

Public Files

The server maintains a directory (see Software on page 13) of public files. Most files can be added or removed from this directory using the usual UNIX commands. Schema files require an additional step to define symbolic links for the purposes of inter-schema references.

The **relink** program will delete all old links and recreate any new links that can be deduced from the schema files in the public directory. **relink** calls **symlink** on each schema file in the public directory. **relink** takes no arguments. **symlink** should be called as follows:

symlink -r *<schema-file>*

The **-r** flag causes resolution processing to be skipped (which presumably cannot occur because the symbolic links necessary for inter-schema references have not been created).

Key-specific Directories

For each existing key, a directory exists to contain all key-specific files. Schema files, Part 21 exchange files and all other user files are stored at the top level in this directory. The directory name is the key itself. A key is represented by an integer. New keys are generated by incrementing the last key and wrapping back to zero if the keys grow large enough. The last key used is stored in the key file in the spool directory (see Software on page 13).

Transactions that take place without a key, are assigned a temporary one. After the transaction is completed, the key and the directory are deleted.

If the key has an associated password, the directory will contain a file called "**.password**" which contains the password. The password is not encrypted. There is no need to encrypt passwords since the server provides no user commands to read plain files.

The e-mail address (and optional user name) that sent the newkey request is stored in the file "**`.creator`**". This is useful if there is a problem with the directory. "**`.creator`**" is not otherwise examined.

Each time a key directory is accessed, the file "**`.last_access`**" is rewritten to contain the name of the current user. This information in itself is not used anywhere, although it could conceivably be useful for debugging. More importantly, the modification date on this file is used to decide how old files are, which is helpful in keeping the system free of files that are no longer in use.

Key-specific files that have not been accessed in thirty days are subject to deletion. An automatic garbage collection occurs whenever the server runs.

All of the per-key files that are used to maintain server overhead information (rather than user information) are prefaced with a "**`.`**". The server does not allow users access to such files, nor do the applications.

File Ownership and Permissions

The files in the directory that contains the user-specific key directories are written by user **`daemon`** for e-mail transactions and user **`express`** for login transactions. To simplify ownership and security, this directory and all files within should be world-writable. This also reduces problems faced by the server administrator while doing manual maintenance of the key directories or the key or log file (which are also stored in the same top-level directory).

Ownership of all other files in the server is irrelevant (assuming they are world-readable) as no other files are written or updated while the server is performing user transactions.

Security

File references by users or indirectly by applications are restricted to public files or files in a key directory. Users can only write or delete files in their own directories. Some applications (e.g., Data Probe) which would otherwise permit arbitrary filesystem references are called with special flags to enforce the aforementioned restriction. Calling sequences are controlled by the server itself which does not permit arbitrary flags to be passed to applications.

Access to user directories may be protected by keys. The use and implementation of this is described elsewhere (see Keys – Storing Multiple Files on page 6 and Key-specific Directories on page 15 respectively).

All files written during operation of the server are maintained in the spool directory. Keeping all writable files separate from the server itself and other static files simplifies the problems in administering access levels to each file. The spool directory is world-writable. All other files and directories in the system are not world-writable. If the spool directory is damaged or destroyed (such as by a bug in the server), it can be restored simply by recreating the directory and initializing the log and key file. While not restoring the user files, this at least restores service to users very quickly. Restoration of user files or of the server itself requires that the entire directory hierarchy be restored from backup.

Modifying the Server or Applications

The server itself may be modified by editing any files in the source directory (see Software on page 13). Executing "**`make`**" in this directory will copy the appropriate files to the public directories.

Application are stored as symbolic links to executables in other directories. This allows the binaries to be shared with PDES Testbed users or developers for testing purposes. "**`make apps`**" copies applications to **`tribble`** in order to isolate any testbed changes from server users who should not be affected by changes to the PDES testbed.

New commands may be added by providing a new procedure and adding it and a corresponding command name to the list of commands accepted by the server.

The server can be tested by running it by hand. Input should look just like an e-mail message. The only input absolutely necessary is a **`From:`** line separated by a blank line before the body of the message. Normally, output is e-mailed back, however the server will produce output at the terminal instead, if the server is run with the flag **`-c "set debug 1"`**. In addition, the application programs will be assumed to come from the current directory and the spool directory is in the relative directory named "**`./spool`**". This is convenient for testing the server from its source directory. A number of other debugging aids are controlled by statements in **`common.exp`** that are normally commented out.

Current Limitations and Future Enhancements

While fully functional, there are admitted limitations in the server, and many enhancements can be made. Applications will be added. Applications which send out binaries could be augmented to support multiple architectures. Applications could send out actual code or object files. This section briefly describes some of the more obvious limitations that may be addressed in the future.

Currently, no attempt is made to derive high-level statistics from the log. This should be done automatically. At the same time, the log file should also be trimmed to only keep a recent history such as one year.

The current access mechanism is simple but effective. A more sophisticated access system may be desirable as more sophisticated collaborations are attempted. For instance, it is probably desirable to reference files from multiple keys at the same time, and differentiate between reading and writing access. Some provision should be made for long-term access to files that do not fit well with the current automatic garbage-collection scheme.

While the command structure is currently consistent, extensibility was not a high-priority goal in designing the server. Future applications or enhancements may require a redesigned command-structure or commands that diverge significantly from others as they exist now. A clear priority in the current server was simplicity to the user and ease-of-use. The command structure will certainly be revisited as new applications or functionality is added to the server.

The server does not follow the Multipurpose Internet Mail Extensions (MIME) [19], a relatively new specification for structured e-mail. MIME would simplify some of the server's user interface such as having to deal with uuencoded, compressed, and split files. Currently however, MIME requires specialized user interfaces which are not in common use. When they are more common, the EXPRESS server should be converted to use MIME.

Acknowledgments

The EXPRESS server was funded by the NIST Scientific and Technical Research Services and is part of the Persistent Object Base Evaluation project. Some of the programs controlled by the server were designed and built as part of the Application Protocol Development Evironment (APDE) project which funded jointly by the Computer-aided Acquisition and Logistic Support (CALS) program of the Office of the Secretary of Defense and ARPA. Development of shtolo was funded by CALS as part of the APDE project.

The author gratefully acknowledges Scott Paisley for significant ideas that become components in the server, and Kent Reed, Barbara Goldstein, Allison Barnard, and Douglas Martin who were very patient and understanding while trying to use early versions of the server.

Thanks to KC Morris and Jim Fowler for project management. Both also provided significant improvements to the content and style of this paper.

Disclaimers

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.

Both the application software and the server software are experimental. No claims are made for either. The software may change unexpectedly as we fix (or add) bugs. Esoteric behavior (such as disk full crises) will probably not ever be handled gracefully.

In no event will NIST be liable for damages, including any lost profits, lost monies, or other special, incidental or consequential damages arising out of the use or inability to use (including but not limited to loss of data or data being rendered inaccurate or losses sustained by third parties or a failure of the program to operate with programs not distributed by NIST) the programs, even if you have been advised of the possibility of such damages, or for any claim by any other party.

References

[1] Mason, H., ed., “Industrial Automation Systems – Product Data Representation and Exchange – Part 1: Overview and Fundamental Principles”, Version 9, ISO TC184/SC4/WG PMAG Document N50, December 1991.

[2] 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.

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

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

[5] Libes, Don, and Fowler, Jim, “The NIST EXPRESS Toolkit – Requirements”, NISTIR 5212, National Institute of Standards and Technology, Gaithersburg, MD, June 9, 1993.

[6] 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.

[7] 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.

[8] Libes, Don, “The NIST EXPRESS Toolkit – Obtaining and Installing”, NISTIR 5204, National Institute of Standards and Technology, Gaithersburg, MD, June 9, 1993.

[9] Libes, Don, “The NIST EXPRESS Toolkit – Using Applications”, NISTIR 5206, National Institute of Standards and Technology, Gaithersburg, MD, June 9, 1993.

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

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

[12] Libes, Don, “The NIST EXPRESS Toolkit – Updating Existing Applications”, NISTIR 5205, National Institute of Standards and Technology, Gaithersburg, MD, June 9, 1993.

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

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

[15] Sauder, D., “Data Probe User’s Guide”, NISTIR 5141, National Institute of Standards and Technology, Gaithersburg, MD, March 1993.

[16] Morris, K.C., “Architecture for the Validation Testing System Software”, NISTIR 4742, National Institute of Standards and Technology, Gaithersburg, MD, January 1992.

[17] Morris, K. C., Sauder, D., Ressler, S., “Validation Testing System: Reusable Software Component Design”, NISTIR 4937, National Institute of Standards and Technology, Gaithersburg, MD, October 1992.

[18] Libes, D., “Expect: Scripts for Controlling Interactive Programs”, *Computing Systems*, pp. 99-126, Vol. 4, No. 2, University of California Press Journals, CA, Spring 1991.

[19] Borenstein, N., and Freed, N., “MIME (Multipurpose Internet Mail Extensions): Mechanisms for Specifying and Describing the Format of Internet Message Bodies”, RFC 1341, Bellcore. Innosoft, June 1992.