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

NISTIR 4334

National PDES Testbed Report Series

QDES Administrative Guide

NATIONAL PDES TESTBED

NIST

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

NISTIR 4334

National PDES Testbed

NATIONAL PDES TESTBED

QDES Administrative Guide

Stephen Nowland Clark

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

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

May 21, 1990

DEPARTMENT OF COMMERCE
UNITED STATES OF AMERICA

Table Of Contents

Disclaimer

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

QDES Administrative Guide

Stephen Nowland Clark

1 Introduction

The Quick-n-Dirty Editor for STEP (QDES) is a prototype editor for STEP product models [Clark90c]. QDES is distributed with the NIST PDES Toolkit [Clark90a]. This document provides an overview of administrative procedures for QDES. It should be read by anybody who will be responsible for installing, maintaining, or upgrading QDES. Two primary administrative tasks are described: making a new release of QDES, and loading a new conceptual schema into QDES. An administrator is often faced with a more nebulous task, as well: namely, to explain curious behavior ("bugs" or "features") encountered by a user. Some aspects of QDES internals, as well as common problems, are described to aid in this task. Knowledge of QDES, the Smalltalk-80™ [Goldberg85] environment, and UNIX™ are all probably helpful in understanding the issues and procedures discussed.

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.

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

1.2 Smalltalk-80 in 60 Seconds

If you don't know Smalltalk-80 at all, here are some points to keep in mind:

- The mouse buttons have non-traditional names in Smalltalk-80. The left, middle, and right buttons are called red, yellow, and blue, respectively. This is an historical artifact: the first Smalltalk machine had colored mouse buttons.

- Most functions in the Smalltalk environment are performed using the mouse: the left (red) button is used for click-and-drag selection, and different popup menus are associated with each of the other two buttons. These menus often vary with the context in which the cursor is located. Thus, general control menus can be popped up on the QDES root, while window- or pane-specific menus are available when the cursor is in a particular window.
- A Smalltalk-80 "program" is saved as a snapshot of the entire state of the environment at some point in time. This snapshot is saved in an image file with the suffix `.im`. The Smalltalk source code (for the system and the application) is contained in a "sources file," with the suffix `.sources`. The sources file is not normally written to during a session. Instead, Smalltalk maintains a "changes file," with the suffix `.changes`, which contains a record of all (most) of the actions taken during the session; these changes can be consolidated and migrated to the sources file periodically. The sources file can be shared by several users simultaneously; each user has his own changes file.
- Because a snapshot file stores the complete state of the system at some point in time, a snapshot taken after an error has occurred can be used to debug the error at a later date. The changes file can also be used to recover from an error; see "Crash Recovery" in [Parc88].

1.3 Trade Secrets

QDES is not perfect. This is not really a secret! Among other things, see the "Known Bugs" section of [Clark90c] and the "Common Problems" section of this document.

There is a hidden "programmer's menu" which can be selected on the QDES root window. To do this, hold down the left-hand shift button while clicking the yellow (middle) mouse button. The programmer's menu pops up, and a selection can be made as from any other popup menu.

2 Architectural Overview

Although a detailed description of the internals of QDES is beyond the scope of this document, in this section we give a quick overview to provide background for the administrator. Although an understanding of this material is not required for the procedures described in section 3, the administrator should find it useful in explaining unexpected features of QDES, and in understanding some of the common problems described in section 4.

2.1 Basic QDES Classes

Every entity class in QDES' schema is represented by a Smalltalk-80 class; the inheritance structure of the schema is mirrored by Smalltalk's class hierarchy. Each entity class is ultimately a subclass of a class called `Pdes`, which is an abstract class with no protocol. `Pdes` inherits from `ExpressAbstract`, which inherits from `Express`. For various practical reasons, two classes are needed at this level, although there is no conceptual difference between `Express` and `ExpressAbstract`. These two classes define protocols which are common to all entities and entity classes; thus, a more appropriate name might be `Entity`. Class variables in `Express` hold the various Browser menus, a `Scanner` for the lexical analysis of user input, the clipboard, and various state information.

One class variable in `Express` (called `Organization`) holds the organizer which the Browsers examine. The concept of an organizer comes from the Smalltalk-80 implementation of Browsers; there, a global instance of class `SystemOrganizer` maintains the list of classes which are visible in a Smalltalk Browser, as well as the groupings of these classes into categories. Similarly, there is class `ExpressOrganizer` which maintains the list of entities known to QDES' Browsers. The actual entity instances in a QDES model are stored in an instance of `ExpressEntityDictionary`. This seeming near-redundancy of function makes it much simpler to reuse the Smalltalk Browser classes in QDES.

2.2 The Express Connection

In order to easily support the creation of new entity classes with appropriate protocol, `Express` defines a special subclass creation method in the protocol `Express class>subclass creation`. This message, which is always sent to class `Pdes`, deals with classes which have no explicit superclass, as well as adding appropriate attribute accessing protocol and instance variable type information to the new entity class.

Fed-X-QDES [Clark90b] reads an Express language [Schenck89] information model and generates code to invoke this special subclass creation method for each entity class in the model. In fact, two output passes are made over the populated working form. The first pass, in (roughly) superclass order, creates all of the entity classes without any instance variables. In the second pass, the entity classes are redefined with instance variables and type information; the (Express) types from a particular schema are generated before the entities from the schema are redefined. This two-pass approach is necessary because Smalltalk quietly accepts forward references to as-yet-unknown classes, and does not handle these references properly when the classes are later defined.

2.3 Dependents and Change Protocols

In order to isolate the data being manipulated from user interface details, Smalltalk-80 uses a paradigm called Model-View-Controller. A Model is a piece of data being manipulated. It knows nothing about the other two components. A View deals with graphical output. It knows about the Model, and queries it to control screen output. A Controller deals with user input. It knows about the Model, and can manipulate it.

A Model has an associated list of dependents. These are objects which have expressed an interest in hearing about changes to the Model. The Model also has a well-defined "change protocol," an indication of the sorts of changes about which its dependents will be notified. When a View is created on some Model, the View adds itself to the Model's dependent list. Whenever the Model changes, it broadcasts a message to all of its dependents, according to its change protocol. Each dependent can then choose for itself how to respond to this notification. So, in a typical interaction, a user will perform some action (mouse click, keyboard input, etc.), which is interpreted by a Controller. The Controller sends its Model a message asking it to modify itself in some way. The Model performs this modification and broadcasts an `update:` message to its dependents, which are Views. The Views wake up and query the Model to see what has changed, and then redisplay themselves accordingly.

This same paradigm is used in QDES. There are two kinds of Models which initiate `changed:` messages: entity instances and the sole instance of `ExpressEntityDictionary`. In addition, the Model of a Browser View is actually an instance of `ExpressBrowser` or one of its subclasses. Each of these Browsers depends on the entity dictionary and on the entity instance which is currently selected. The various subviews of a Browser View also depend on the Browser, and the individual field Views depend on the selected instance. Figure 1 shows the full dependency graph for a QDES session with a single System Browser whose current selection is a date.

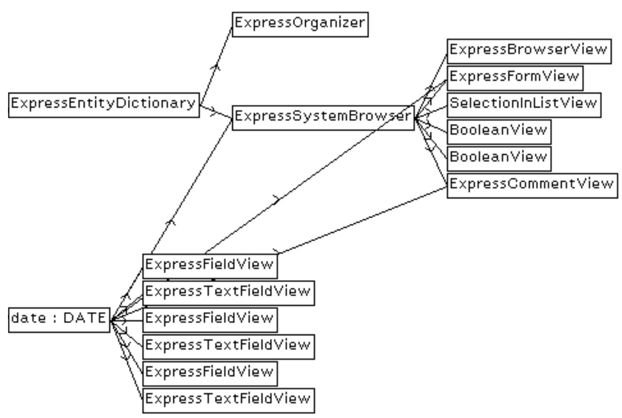

Figure 1: QDES Dependency Graph

Let's look at a couple of examples to see how changes get propagated.

First, suppose we change the value of the `month` attribute of our date. The date object is eventually sent the message `fieldValueAt: 'month' put: someInteger`. This method does some finagling, puts the value into the right slot, and then calls `self changed: #value with: 'month'`. The `changed:with:` message initiates a change broadcast. `#value` indicates the "aspect" of the Model which has changed. `'month'`, which is an optional parameter in the change protocol, will help the various dependents determine whether they are interested in this particular change. Looking at the "updating" protocol for `ExpressSystemBrowser`, `ExpressFormView`, `ExpressCommentView`, and `ExpressFieldView` (which represents the field labels in an entity form), we find that these objects all ignore updates of the `#value` aspect. Thus, only the

ExpressTextFieldViews (which display actual attribute values) will respond to this broadcast, using its update:with: method. Each of these views keeps an indication of the name of the attribute which it views, and will be interested in the broadcast only if its with: parameter matches the view's own field name. Thus, although nine objects receive the broadcast, only one ends up doing anything about it: the view which is displaying the value of the month attribute.

Now, suppose we create a new entity instance. Eventually, the entity dictionary receives the message defineEntity: with some entity instance as a parameter. It adds this entity to itself, and then sends itself the message changed: #organization. The dictionary has only two dependents. When the ExpressOrganizer is sent update: #organization, it rebuilds its list of entities from the dictionary. (As an aside, note that this is the reason that a file load takes exponential time under normal circumstances: after each entity is created, the organizer re-sorts its list. Turning off the display update temporarily removes all of the entity dictionary's dependents, so that this sorting is not done). The Browser Model ignores updates on this aspect. The second change broadcast from the dictionary, on aspect #list, is ignored by the organizer but not by the Browser, which turns around and broadcasts changed: #category to its dependents, the various subviews of the Browser View (#category is a carry-over from the Smalltalk-80 Browser classes). The SelectionInListView, which displays the Browser's selection pane, responds by querying the organizer for the new entity list. The #sameSelection parameter that comes with the dictionary's broadcast indicates that the current selection has not changed. Otherwise, the ExpressFormView would respond to this change by rebuilding itself from the new selection. The change protocol must use two separate aspects (#organization and #list) to announce the same change in order to ensure that the organizer updates itself before the Browser views query it to rebuild themselves.

3 Creating New QDES Images

There are two situations in which it becomes necessary to build a new QDES image: when a new version of the editor is ready to be released, and when a new schema (or a new version of an old schema) needs to be imported into QDES. The procedure for the latter is somewhat more involved, and so we address the former first. Throughout these procedures, the root directory of the QDES distribution is referred to as ~qdes/. In the current Toolkit distribution, this directory is ~pdes/applications/qdes/.

There is a working directory, ~qdes/scratch/ which is intended to be used as a workspace for building new images. Both release procedures begin in this directory. For the sake of example, assume that QDES is currently at version 2.2, so that the root name for the three image files is Qdes-2.2. (Note: for brevity, ~qdes/ is used throughout this document to refer to the root directory of the QDES distribution).

A QDES image contains two Projects, or desktops. A user only sees one of these, the QDES Project. The programmer's Project contains the normal Smalltalk-80 programming environment, with code Browsers, Workspaces, etc. In particular, it contains two important workspaces, collapsed to icons labelled "New Schema" and "New Version." These workspaces contain Smalltalk scripts used in the QDES release procedures. In the programmer's project, the QDES project appears as an empty window labelled "QDES Project." To switch from the programmer's to the QDES Project, select "enter" from this icon's yellow (middle) button menu. To switch back to the programmer's Project, select "smalltalk" from the hidden programmer's menu.

Most of the work in creating new QDES images is performed by well-tested scripts and simple UNIX commands. Nonetheless, problems may occasionally arise. These problems, particularly UNIX errors, should be addressed immediately, as they can have serious repercussions later on. If any of the UNIX commands reports an error, STOP. Figure out what's wrong, and correct it before proceeding. If something fails while you're in Smalltalk, save the image *under a different name* (pick "save" from the programmer's menu, and type a new name into the Prompter which appears) before exiting. Otherwise, the time you've spent in Smalltalk so far is, for all intents and purposes, lost.

In some cases, it is possible to modify an existing image in place. The basic rule is: if you make no changes which are logged to the changes file (for the most part, method compilations and fileIns are all that are logged), you needn't do most of the work described in the following sections. Examples of useful changes which are not logged include rearranging the screen layout (including building new windows) and changing the state of QDES, for example by turning off display update. In these cases, it is sufficient to start up QDES in the scratch directory (or elsewhere outside of `~qdes/`), make the changes, and resave the image *with the same name*. Then exit Smalltalk, make sure nobody is using the image you've modified, and `mv` the new `.im` and `.changes` files into `~qdes/`.

3.1 Making a New Release

This section describes the procedure for building a new release of QDES. The end result of this procedure will be a virgin image with no schema loaded, called `Qdes-2.3`. All Smalltalk menu selections mentioned are from the yellow (middle) button menu, unless otherwise indicated.

- In `~qdes/scratch/`, extract the contents of `Qdes-2.2.tar.Z`, the current virgin QDES release:

```
% uncompress Qdes-2.2.tar.Z
% tar xvf Qdes-2.2.tar
```

- Make symbolic links to `scratch/Qdes-2.2.sources` and `scratch/Qdes-2.3.sources` in `~qdes`:

```
% pushd ..
% ln -s scratch/Qdes-2.2.sources
% ln -s scratch/Qdes-2.3.sources
% popd
```

  Both of these links are required, since QDES looks for its sources file in `~qdes/`, and will look for both the old and the new sources file in the course of the session.

- Start up the image:

  ```
  % /usr/smalltalk/bin/st80 Qdes-2.2.im &
  ```

  QDES will start up in the (normally hidden) programmer's Project. Make your changes. This may be done in two ways: a developer may hack directly on this image, or an administrator may have available Smalltalk-80 source code provided by a developer. In the latter case, the source code must be filed in using a FileList, available from the Smalltalk root menu. To do this, pick "file list" from the Smalltalk root menu (the middle-button menu on the background) and frame the new window when prompted. Assuming the source code to be filed in is in the current directory, type `*.st` into the top section of the file list, and pick "accept," again from the middle-button menu. A list of files will appear in the next pane. Select `mySchema.st` from this list and pick "file in" from the middle-button menu. Close the file list when the fileIn is complete (pick "close" from the right button menu).

- Open the "New Schema" workspace (click the left button on the box labelled "New Schema"). Change all occurrences of `Qdes-2.2` to `Qdes-2.3` in this workspace. This will ensure that everything works properly when a schema is later loaded into this new version. Collapse this workspace after making these changes.

- Look at the "New Version" workspace. It contains a script of the Smalltalk-80 commands needed to build the new image. Change all occurrences of `Qdes-new` in this workspace to `Qdes-2.3` (the new version name). Select the entire contents of the workspace and pick "do it." This will build `Qdes-2.3.sources` and save the new image as `Qdes-2.3.im`.

- Quit from Smalltalk.

- Make `Qdes-2.3.tar.Z`:

  ```
  % tar cvf Qdes-2.3.tar Qdes-2.3.{im,changes,sources}
  % compress Qdes-2.3.tar
  ```

- Clean up and install the new image. If you're brave and don't want/need to keep the old version at all, you can `rm Qdes-2.2.tar`; otherwise, `compress Qdes-2.2.tar`. Then:

  ```
  % rm Qdes-2.2.{im,changes,sources}
  % rm ../Qdes-2.2.sources ../Qdes-2.3.sources
  % mv Qdes-2.3.{im,changes,sources} ..
  ```

3.2 Loading a New Schema

This section describes the procedure for loading a schema (called mySchema for the sake of example) into QDES 2.2. The end result of this procedure will be a QDES image called `Qdes-2.2-mySchema`. If you are replacing an old image called `Qdes-2.2-mySchema` (i.e., this is a new version of an existing schema) make sure that nobody is using it before proceeding. All Smalltalk menu selections mentioned are from the yellow (middle) button menu, unless otherwise indicated.

- Push `mySchema.exp` through the Fed-X-QDES translator (for more information on Fed-X-based Express translators, see [Clark90b]) to get the QDES fileIn, and put the result into the QDES scratch directory:

```
% fedex_qdes -e mySchema.exp -o mySchema.st
```

- In `~qdes/scratch/`, extract the contents of `Qdes-2.2.tar.Z`:

```
% uncompress Qdes-2.2.tar.Z
% tar xvf Qdes-2.2.tar
% compress Qdes-2.2.tar
```

- Go to `~qdes/`:

```
% pushd ..
```

- If you are replacing an old image, move its sources file aside:

```
% mv Qdes-2.2-mySchema.sources
     Qdes-2.2-mySchema.sources.aside
```

  This will save the old sources file, so that you can back out to the old version of the schema if something goes wrong.

- Make symbolic links to `scratch/Qdes-2.2.sources` and `scratch/Qdes-2.2-mySchema.sources`:

```
% ln -s scratch/Qdes-2.2.sources
% ln -s scratch/Qdes-2.2-mySchema.sources
```

  Both of these links are required, since QDES looks for its sources file in `~qdes/`, and will look for both the old and the new sources file in the course of the session.

- Return to `~qdes/scratch/`:

```
% popd
```

- Start up QDES:

```
% /usr/smalltalk/bin/st80 Qdes-2.2.im &
```

  QDES will start up in the (normally hidden) programmer's Project.

- File in the schema to be loaded. To do this, pick "file list" from the Smalltalk root menu (the middle-button menu on the background) and frame the new window when prompted. Type `*.st` into the top pane of the file list window, and pick "accept," again from the middle-button menu. A list of files will appear in the next pane. Select `mySchema.st` from this list and pick "file in" from the middle-button menu.

- Get a cup of coffee. Read the paper. The loading process can take as long as several hours. Leave the mouse in the middle pane of the file list; when the schema is finished loading, the scrollbar on this pane will reappear. When this happens, close the file list (with the right-button menu).

- Open the "New Schema" workspace (click the left button on the box labelled "New Schema"). Change all occurrences of `Qdes-2.2-schema` in this workspace to `Qdes-2.2-mySchema`. Select the entire contents of this workspace and pick "do it" from the middle button menu.

- Get some more coffee (but not as much as before). Again, the scrollbar will reappear when Smalltalk is ready to proceed, provided that you leave the mouse inside the workspace. Collapse this workspace (pick "collapse" from the right button menu).
- Enter the QDES Project (pick "enter" from its yellow button menu), and set up the screen layout to taste. This might involve creating certain clas hierarchy views which you would like always to be available, positioning the Feedback window, and/or creating an initial System Browser,
- Save the image (pick "save" from the hidden programmer's menu). Call it `Qdes-2.2-mySchema`. This will be the default value provided when you are prompted for a name, so you can just hit <return>.
- Quit from Smalltalk/QDES.
- Clean up and install the new image.

```
% rm Qdes-2.2.{im,changes,sources}
% rm ../Qdes-2.2.sources
% rm ../Qdes-2.2-mySchema.sources
% mv Qdes-2.2-mySchema.{im,changes,sources} ..
```

  If you are replacing an old image, you should also

```
% rm ../Qdes-2.2.sources.aside.
```

  For safety, you should probably explicitly set the read-write permissions on the new image files. The convention has been for the `.sources` file to be read-only, and the `.im` and `.changes` files to be writable only by the owner:

```
% cd ..
% chmod 644 Qdes-2.2-mySchema.{im,changes}
% chmod 444 Qdes-2.2-mySchema.sources
```

- Create a script in `~pdes/bin/` to run the new image. Call it `qdes-mySchema`. Refer to the template in `~pdes/bin/qdes-template`.

4 Common Problems

In this last section, we briefly discuss several problems which QDES users unfortunately encounter fairly often, and which an administrator is therefore likely to be asked to address.

- Sometimes Browsers don't update themselves properly. This may be due to various known problems or due to some mysterious conjunction of astronomical bodies. In any case, the first thing to try is to pick "update browser" from the selection pane menu in any affected Browser; if this doesn't solve the problem, try a new Browser.
- The bug described above usually has to do with objects vanishing from the entity dictionary's dependents list. More often than not, it is the organizer which is lost. Remember that Browsers base their view of the world on the contents of the organizer. Thus, even though they know that things are changing, they can't see these

changes as long as the organizer is not being updated. The workaround is to add the organizer back into the dictionary's dependents list, and then to send the dictionary `changed:` messages to get the world back in sync:

```
Express entities addDependent: Express
     entityOrganization.
Express entities changed: #organization.
Express entities changed: #list with:
     #sameSelection.
```

This code can be executed in the "Bug Workaround" window or in a workspace created from the programmer's menu. Occasionally, a Browser itself will disappear from the dependents list. The simplest workaround for this is to get rid of the offending Browser and execute the above fix. New Browsers will behave properly.

- Particularly when running a virgin QDES image, it sometimes happens that attempting to browse QDES source code or to execute one of the image release scripts discussed above results in an error notifier appearing with the message `Unhandled exception: File/directory does not exist`. This usually means that Smalltalk cannot find its sources file. Since the sources file is shared and not writable, it has an absolute path. QDES looks for it in `~qdes/`. This is why it is necessary to create the two symbolic links in `~qdes/` before trying to build a new image. Make sure that any links created are in the proper directory, are spelled correctly, and point at the proper files. In extreme circumstances (e.g., if the sources file has actually been deleted), it may be necessary to back out the a virgin QDES image and reload the schema.
- It is possible, particularly in older QDES images, for the entity class hierarchy menu to be utterly wrong. This happens when the menu has been built in a previous image, from a different schema, and never rebuilt from the current schema. The fix for this is to pick "initialize express" from the programmer's menu. Since QDES assumes that this indicates that a new schema has just been loaded, any instantiated objects in the workspace will be deleted. This initialization is now done automatically by the script to load a new schema, so the problem should vanish rapidly.
- Although this is not immediately obvious, Smalltalk-80 runs inside a Sunview™ window when running on a Sun Workstation. This window can be manipulated just as any other Sunview window can. Combined with Smalltalk's fairly slow reaction to graphical input, this can cause some surprising results. For example, a user might accidentally select something from the Sunview menu at the window border, perhaps collapsing Smalltalk to an icon ("Where did it go, and what's that funny box that says st80?"). Sunview and Smalltalk refreshes may get out of sync, leaving part of the screen blank and inaccessible to Smalltalk. A couple of Sunview refreshes should clear this up.
- There has been a persistent bug in the QDES intermediate file fileOut code. I believe it has finally been eradicated, but one never knows. Older versions of the Smalltalk-80 compiler required entity names in quotes (that is, as attribute values) to be preceded by `#`. This is no longer necessary, and is, in fact, an error. If a QDES fileOut file complains about "improper syntax for entity" when filed back in, this is

probably the problem. The workaround for existing QDES files is to globally replace `'#` with `'`. That is, replace quote-hash with quote. The fix in QDES is to make sure that the method for `Express|wfString` reads

```
wfString
     ^self entityName asString printString
```

A References

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

[Goldberg85] Goldberg, A. and D. Robson, Smalltalk-80: The Language and its Implementation, Addison-Wesley, Reading, MA, July, 1985

[Parc88] ParcPlace Systems, The Smalltalk-80 Programming System, ParcPlace Systems, Mountain View, CA, 1988

[Schenck89] Schenck, D., ed., Information Modeling Language Express: Language Reference Manual, ISO TC184/SC4/WG1 Document N362, May 1989

[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

NATIONAL PDES TESTBED

MAIL TO:

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

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