dynamicxport

User Guide

Revision 2.2

(Product release 2.2)

01/19/2004

powered by dL4

Information in this document is subject to change without notice and does not represent a commitment on the part of Dynamic Concepts, Inc. (Dynamic). Every attempt was made to present this document in a complete and accurate form. Dynamic shall not be responsible for any damages (including, but not limited to consequential) caused by the use of or reliance upon the product(s) described herein.

The software described in this document is furnished under a license agreement or nondisclosure agreement. The purchaser may use and/or copy the software only in accordance with the terms of the agreement. No part of this manual may be reproduced in any way, shape or form, for any purpose, without the express written consent of Dynamic.

Dynamic Concepts, Incorporated

18-B Journey

Aliso Viejo, CA. 92656

www.dynamic.com

Table of Contents

General

Product Definition

This product provides an API (Application Program Interface) toolkit that allows software developers to integrate a Web server to a Unibasic or Dl4 database. This product facilitates the access to real-time data, such as inventory availability, order status, and account balances.
Using this toolkit you can provide your (or your client's) customers, suppliers, and employees easy Internet browser based access to your information. The API toolkit let's you seamlessly integrate your data to your Web site.

DynamicXport also has many additional components built on it's own architecture to provide commonly needed functionality. These components include user name and password authentication, menu option assignment and presentation, send e-mail and credit card processing.

Utilizing HTML Style Sheets, a Session profile and several menu templates the look and feel of the web site is easily customizable and can be configured to different looks for different groups of users.

Custom development is expedited with the RAD (Rapid Application Development) toolkit that is included. The RAD toolkit includes code samples, subroutines and templates and a debugging tool.

Using the DynamicXport platform you can provide secure, browser-based thin client real-time access to your existing database.

DynamicXport can be licensed under several models.
- Developer Kit, not to be deployed for live commercial applications
- Application Server License is an unlimited user run-time license. Development is not available with this license.
- Developer License is an unlimited user license, with the ability to also develop and add new options
- DynamicXport Lite is identical to Application or Developer License except it is limited to the number of user logins that can be created and user logins are limited to one active IP number at any given time.

DynamicXport is an n-tier architecture consisting of :
- A web presentation layer which can be managed by a Web Designer
- A communications layer using socket technology
- A business rules layer where applications are typically written in dL4 or Unibasic
- A database layer which is typically a supported dL4 or Unibasic database

The product consists of the following components:

- ✓ Dl4 license for the DynamicXport server (Windows or Unix) – this is a standard version of dl4, but a special license to be used for web page integration, licensed on a per client/unlimited users basis.

- ✓ If the more secure socket model of outside server/inside listener is implemented an additional Dl4 license (or additional user licenses) is needed for the inside DynamicListener on the application server.

- ✓ Developed using HTML, Javascript and dL4 so it is platform independent. It can run on Unix or NT servers.

- ✓ DynamicXport is compatible with any web server on Unix or NT.

- ✓ CGI command & socket library – consists of dL4 libraries of standard subroutines that reside on the DynamicXport server and listener to facilitate the interface from the Web Server to the database.

- ✓ DynamicListener - the listener component waits on the application server for socket requests coming from the DynamicXport server. The listener then spawns the proper custom dl4 External Program or Unibasic Program based on the request.

- ✓ Sample Dl4 and Unibasic programs are provided for reference and to be used as templates for development.

- ✓ Server to Server VPN security option – this component allows the DynamicListener to reside on a separate box inside a firewall and talk to the outside DynamicXport server using VPN encryption.

- ✓ Browser to Server SSL security component option – this component allows the browser to communicate to the Web Server using a secure socket layer.

- ✓ Username/password encryption & authentication module –
  - Tiered Administration functions:
    - Organization definition and administration
      - Allows client to setup customers as organizations with a designated administrator who can setup user logins and options through the browser
    - User ID and Password administration
  - Allows user to change their own password.
  - Allows Group Administrators to add/modify users in their group.
  - User authentication at each application request
  - Support of cookies to save Session ID at client PC, if desired (not implemented)
  - Session ID encryption and state maintenance
  - IP Verification and session timeout options

- ✓ User options module –
    - Menu Options definition and administration
  - Dynamically generated HTML Menu Options display
    - Several predefined option interfaces to choose from or customize the interface with HTML and standard dL4 list and table tags.
    - Each organization (customer or group) can have a different customized interface

- ✓ Credit Card module
  - Credit cards can be securely processed for payment through an API to Verisign Payflow Pro.

- ✓ Upload/download module
  - Allows users to upload files to the web server or directly to the application server.
  - Allows users to download files directly from the application server.

- ✓ SimpleInterface –

  - Allows application to receive Form input using a simple Call Routine to retrieve each fieldname into a variable.
  - Allows application to output form field values using a simple Call Routine which sets a fieldname equal to a value.
    - These fields would be interfaced into the html output document wherever the fieldname is found.
  - The SimpleInterface allows a developer to create the backend custom programs in a pure Unibasic environment.

- ✓ Fully functional code samples and templates - includes HTML and dL4 code to help you expedite development efforts.

- ✓ Documentation, instructions and examples on how to implement the APIs.

Terminology

Web Server, Outside Server, DynamicXport Server are all synonymous and refer to the internet or intranet web page server machine. It is the machine that stores the HTML pages and the CGI to communicate through a socket to the Application Server.

Application Server, Inside Server, DynamicXport Listener are all synonymous and refer to the machine that stores the Application programs and database. It is also the machine that stores the socket listening program to respond to web requests.

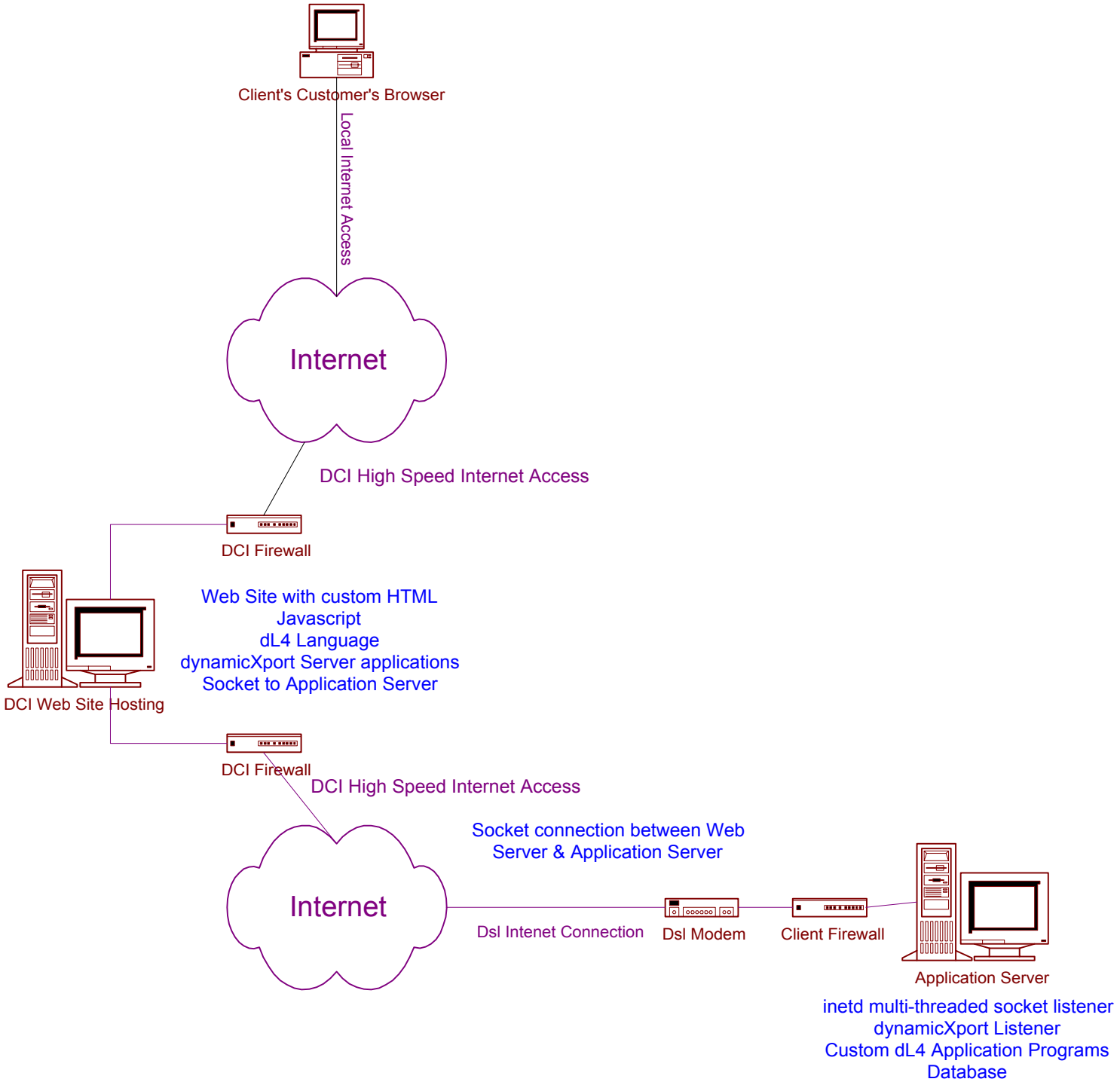
Topologies

Many network topologies and security mechanisms can be used to implement internet integration. Choosing the appropriate topology is based on the location of the data, the sensitivity of the data and the budget available.

Topology #1 : Single Server : Least expensive Internet or Intranet
The DynamicXport Server and Listener components are loaded onto the Application/Web server and does all it's processing on a single server. The server that contains the dl4/Unibasic application is configured to be a Web server. This configuration is an easy way to implement an intranet. The security risks associated with passwords, applications and data stored on a server accessible through the internet makes this a higher risk topology for an internet application.

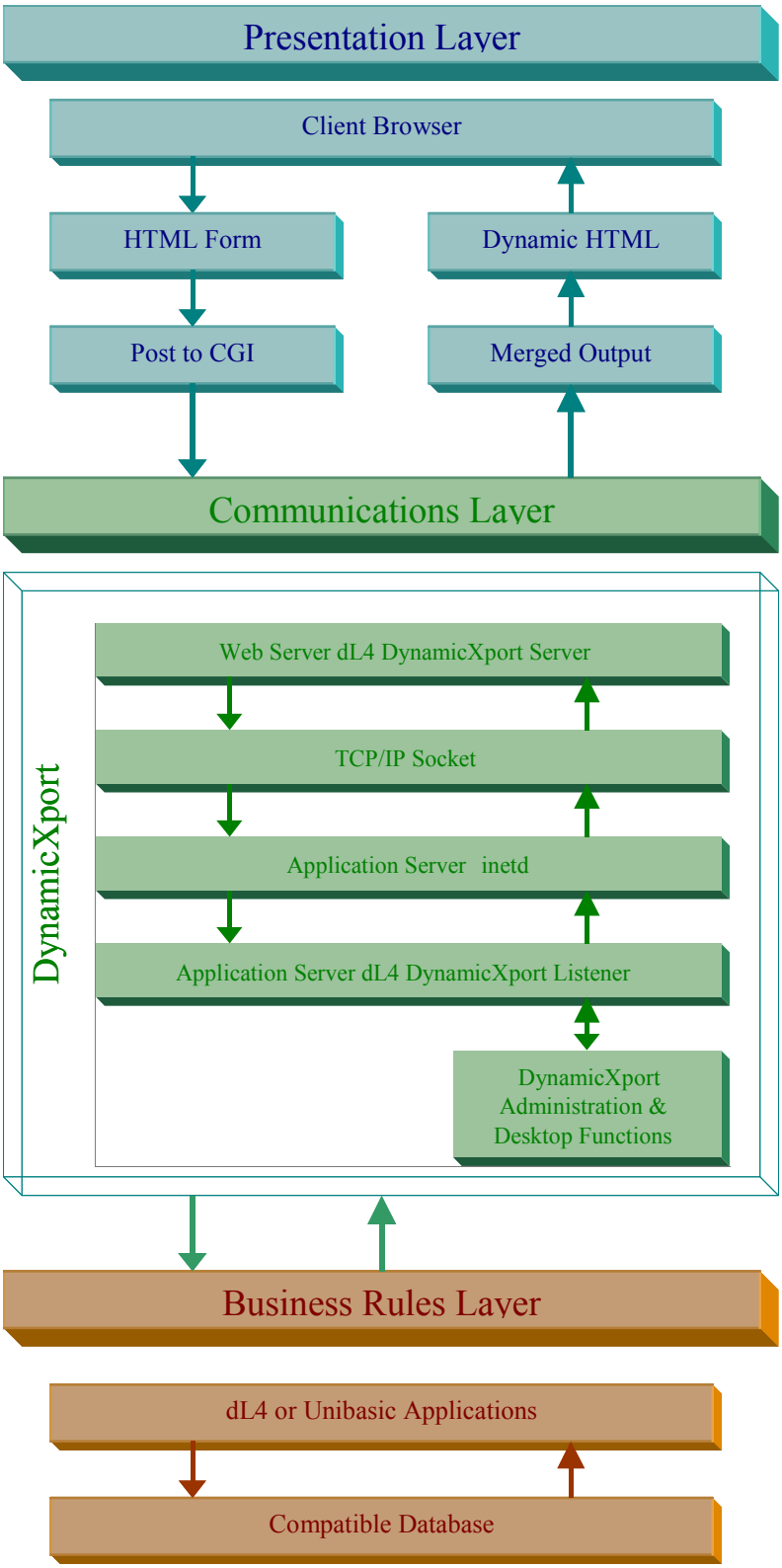
Topology #2 : Two servers : Web server on outside of firewall, Application server on inside of firewall.
The DynamicXport server is a separate machine which is located outside the firewall, while the DynamicListener is loaded on the application server on the inside of the firewall for password and data protection. Communication is performed through a TCP/IP socket connection between servers. The firewall is configured so that only the Web server IP address is allowed into the TCP/IP socket.

Topology #3 : Two servers : Remote Web server at a hosting service , Application server at another location connected to internet and protected by a firewall.
This would be the typical topology.
The DynamicXport server is a separate machine which is located at a web hosting service, while the DynamicListener is loaded on the existing application server on the inside of a firewall for password and data protection.
The application server is connected to the Internet with a DSL or similar high-speed static IP connection or for utmost security and speed a private dedicated line. Data passing between servers through the socket could be encrypted through a Firewall VPN. Communication is performed through a TCP/IP socket connection between the servers. The firewall is configured so that only the Web server IP address is allowed into the TCP/IP socket.



Architecture

n-tier architecture



Security

There is a user file in the DynamicXport database on the application server with a User ID and Password.

When a user signs in on the login screen their browser sends the entered User ID & Password to the web server (preferably on an SSL page)

The web server then sends the user's IP Address, User ID and Password to the application server for validation against the User ID and Password file.

If the User ID & Password entered is validated, the user is assigned a new SessionID.
(A valid beginning and ending IP can be also be specified and validated by Organization.)

You can consider the sessionID to be the user's access key as long as they have not exceeded the SYSTEM.Timeout, SYSTEM.Elapsed and SYSTEM.Expired parameters as defined in the SYSTEM file.

No passwords or files are ever stored anywhere on the webserver.

A setting in the SYSTEM file defines the time period allowed between accesses before requiring a revalidate to continue a session. I.E. 15 minutes (SYSTEM.Timeout$)
Another setting in the SYSTEM file defines how long the session is allowed to exist before requiring a revalidate to continue a session. I.E. 24 hours (SYSTEM.Elapsed$)
A setting in the STSTEM file defines how long of an inactivity period is allowed before session is forced to expire. I.E. 8 hours (SYSTEM.Expired$)

Before an Application Server option is run, it will re-verify it is on the user's option access list.

Recommended additional security :

Client to server :

- Login page should always use SSL encryption at minimum
- Other pages should use SSL to protect data and protect session ID

Webserver to Application Server communications :

If the web server is remote from the application server, your options are :
(To protect new user passwords/changed passwords and data being passed from being seen or intercepted)

a) A dedicated private line instead of public internet transport
b) A firewall VPN to encrypt data between servers

Browser Compatibility

DynamicXport is developed and tested to be compatible with Microsoft Internet Explorer (IE) versions 4.0 and later (however 5.0 and later is recommended if the use of stylesheets for menu frames is desired) and Netscape Navigator, versions 4.04 & later (known problems with SSL warnings in Netscape 6.01).

DynamicXport utilizes Javascript, CSS 1.0, Frames and DHTML within the client's browser. The client must have these features enabled in their browser.

Web Server Compatibility

DynamicXport is developed and tested to be compatible with IIS on NT and Windows 2000, and Netscape Fasttrack and Apache Webservers on Unix/Linux.
Other Webservers would most likely work but are not supported.
DynamicXport does not require any licensing or special Dynamic software to be installed on the web server.
Standard SSI is NOT used and NOT required because of various server incompatibilities.
FrontPage extensions do NOT need to be installed on the server.

Application Server Compatibility

DynamicXport is developed and tested to be compatible with various Unix,Linux and Windows platforms.

DynamicXport may function only as a one user demo on Windows XP/98/ME application servers, if some third party wireless networking incompatible software is also installed. A special listener application is provided and must be launched to provide one user demo functionality.

Development

Web Designers Brief

The purpose of this brief is to provide instructions for a web designer to develop forms and results pages that would integrate to a Unibasic or dL4 database using the DynamicXport platform.

A basic understanding of DynamicXport is necessary, in particular it's directory structure.

Overview :

DynamicXport components will be installed on the web server under the site's root directory.
All components of DynamicXport will be contained in directory dx/.

DynamicXport provides the components necessary to provide certain core functionality as follows :

- User login and authentication.
- Users defined and grouped into Organizations.
- Users' menu option pages which are dynamic based on the User's defined option list.
- Links to the proper HTML pages when an option is chosen.
- CGIs to accept form inputs, send inputs through a socket to a local or remote application server for processing by Unibasic or dL4 and send results to the proper output HTML page.
- Each Organization (and it's members (users)) can be defined to receive a different 'VIEW', which dynamically changes the directory in which to find the associated Option Menu Style, Menu Icons, Stylesheet, HTML pages and images.

Rules

Rules to live by :

Rule #1 Do NOT modify any HTML files that begin with ‘DX’.
Rule #2 If you wish to make changes to other HTML files, images, icons, stylesheets provided, do NOT change anything in dxstd subdirectories or dxstd filenames. Instead copy them to a new ‘theme’ and then change.

Copy dxstd and login/dxstd to new directories.
Copy styles/dxstd.css to a new .css filename.
Create a new VIEW record to use the new ‘theme’.

Rule #3 For web-server cross-platform compatibility use the dl4v(S_Root), dl4v(S_MergeDIR) and dl4v(S_OptionDIR) merge variables to define image source and page link directories, with ALL HTML code being at the same directory level as CGI-BIN. CGI-BIN is normally at virtual root/dx/cgi-bin. So HTML should be located in virtual root/dx/custom directory for example and referenced as custom/
(The reasoning for this is that when running CGI some web servers place you in the local directory, virtual root directory or CGI-BIN directory.)

Directory Structure

(where things are and where to put things)

All DynamicXport files are located in the dx/ directory.

The directory structure, in association with a user's 'VIEW' definition, is designed to allow different themes to be defined and accessed by different users. Different themes directories can be independently defined for the Login function, the menu option presentation function and the custom application. The system is installed with the theme 'dxstd' as the default theme. Under each function additional themes can be defined and then linked to by the 'VIEW' definition file (Each Organization is then linked to a 'VIEW' that it's users will receive.).

All files related to the login function are in the dx/login directory, under the default 'dxstd' theme subdirectory. Thus all the login function files for theme 'dxstd' are in directory dx/login/dxstd/.
If you wish to change the appearance of the login function it is recommended that you create a new theme by creating a new subdirectory within dx/login, copy the files from another theme (such as 'dxstd') and then making your HTML or image changes. Your hyperlink to login can then be changed to link to the new theme. (It is OK to change files in the dxstd directory, but not recommended)

Directories are broken down into the following functionality :

dx/cgi-bin You should not modify any of the cgi-bin files (with the exception of dxserver.txt which may need some initial installation changes)

dx/admin You should not modify any of these files. These files are used for the presentation of the DynamicXport administration functions.
This directory has a /images subdirectory containing images used in Admin functions.

dx/login /theme/ All files related to the login function. Since the appearance of the login screen is typically changed it is separated into its own directory with ALL it's related files. To change the appearance of the login create a new theme as defined above and link to it. (You may want to remove or change the link to the default stylesheet)

dx/dxstd/ HTML files related to presenting user menu options, messages, validation and logoff messages. Several standard menu option presentations are pre-configured. If you need to create a custom menu presentation it should be named optionx.html and placed in this directory.
The 'VIEW' definition should also be changed to use Option Type 'X'.

If you need to modify the HTML code in the option files you should create a new theme in the dx/ directory, copy the files from the other theme and then make your modifications. The 'VIEW' definition should also be changed to link to the new theme or a new 'VIEW' definition created. (It is OK to change HTML files in the dxstd directory, but not recommended)

Images associated with Options presentation are stored in the subdirectory /images under /dx/dxstd. (DynamicXport User and User's Options Maintenance pages also use these images) It is OK to replace these images to create a custom appearance. (Originals of the dxstd images are also kept in the dx/admin/images directory. These should NOT be changed as they are used by DynamicXport Administration pages.)

Each option can have a small and a large image associated with it. The option images are stored in the subdirectory /icons under /dx/dxstd. The filenames for option icons MUST be 'optionID'.gif for the small icon and 'optionID'img.gif for the large image. You should create and add the small and large image files for each custom option created and place them in this directory.

dx/styles .css files for defining Styles are stored in this directory. The dxstd.css file is used for defining font and color styles.
If you need to modify Stylesheet parameters you should copy the dxstd.css fileto a newtheme.css file and then make your modifications. The 'VIEW' definition should also be changed to link to the new stylesheet or a new 'VIEW' definition created. (It is OK to change dxstd.css, but not recommended)

dx/includes This directory contains Include HTML and Javascript routines which you are free to use within the custom application as well.

dx/examples This directory contains example pages that are tied to example options.

Dynamic Merge commands

The following are valid commands that can be embedded into an HTML page. DynamicXport will interpret these commands and merge field values into the HTML page before presenting the page to the browser.

dl4v(fieldname) Merge string value

DynamicXport recognizes this as a reference to an output merge field and replaces it with the string value of the associated variable.

HTML example :

```
<body>
Your name is dl4v(name)
</body>
```

DynamicXport will replace dl4v(name) with the value of the field called 'name' which was sent from the application server using the Call DXSET command.

dl4t(fieldname) Merge multiline text box

For simple string values, use dl4v. Output fields can also be multiline text areas. To output multiline text areas the application server would create an output list using the Call DXSET command's "l" (list) type and a string array. DynamicXport will replace dl4t(fieldname) with the values stored in the list of the same field name.

HTML example :

```
<body>
Address :
<textarea>dl4t(address)</textarea>
</body>
```

DynamicXport will replace dl4t(address) with the values stored in the list of the field called 'address'. (Note: dl4t is not routinely used and it's function could also be accomplished by using inline mnemonics for
 in a dl4v field)

The dl4t command should be used in place of the dl4v command if you are merging in long strings that may be more than 1000 characters merged into a single html line.

dl4l(fieldname) Merge list of string values

Any line in HTML that contains a list reference (dl4l) will be printed once for each value in the fieldname output list. To output lists the application server would create an output list using the Call DXSET command's 'l' (list) type and a string array. DynamicXport will replace dl4l(fieldname) with the values stored in the list of the same name and repeat the HTML line. The dl4l reference is typically used to create form <select> boxes.

HTML example :

```
<body>
<form>
<select>
<option>dl4l(choices)</option>
</select>
</form>
</body>
```

DynamicXport will replace dl4l(choices) with the values stored in the list of the field called 'choices' and repeat the HTML line for each instance.

Hidden values :

Using the Call DXSET command's "h" (hidden) type, an application can set hidden values which will automatically be added into all forms on the HTML page. In addition, DynamicXport automatically adds hidden values for the reserved fields 'Session' and 'Option'.

dl4o(nohidden) Do not add any hidden values

If the hidden values should not be merged into a section of the HTML, such as script routines, the dl4o(nohidden) command should be issued to turn the add hidden values function off. To issue the command in HTML it should be contained in an HTML comment, i.e. <!-- dl4o(nohidden) -->

dl4o(hidehere) Add any hidden values here

Complements the dl4o(nohidden) command to tell DynamicXport to add the hidden values at this particular location of the HTML page. To issue the command in HTML it should be contained in an HTML comment, i.e. <!-- dl4o(hidehere) -->

(If the dl4o(nohidden) command is not issued, DynamicXport will automatically add hidden values to all forms on the HTML page without the necessity of any special commands.)

Tables :

DynamicXport facilitates merging into tables by having the application create lists for each column using the Call DXSET command's 'l' (list) type and then merging the lists into an HTML table definition surrounded by dl4c(for), dl4c(next) commands.

dl4c(for) dl4c(next) dl4a(fieldname)

Place the dl4c(for) command at the beginning of the HTML table definition, contained in an HTML comment, i.e. <!-- dl4c(for) -->.

Place the dl4c(next) command at the end of the HTML table definition, contained in an HTML comment, i.e. <!--dl4c(next)-->.

Within the table use the dl4a(fieldname) command to indicate where each list array should be merged. Note that the separate columns of a table will use different field names and thus separate, but parallel list values. "dl4v(fieldname)" references can be used in the repeated HTML code for constant values that do not vary between rows.

HTML example :

```
<body>
<table>
        <!-- dl4c(for) -->
        <tr>
                <td>dl4a(Date)</td>
                <td>dl4a(Invoice)</td>
                <td>dl4a(Amount)</td>
        </tr>
        <!-- dl4c(next) -->
</table>
</body>
```

DynamicXport will replace dl4a(Date), dl4a(Invoice) and dl4a(Amount) with the values stored in the list of the field called 'Date', 'Invoice' and 'Amount' respectively and repeat the HTML code between the dl4c(for) and dl4c(next) for each instance.

Advanced Topic: Extended formats of dl4c

If the format of the dl4c(For) structure is extended to be dl4c(For,*fieldname*), referencing the fieldname as dl4v(*fieldname*) within the for,next loop provides access to the current array index number. This provides a mechanism to number the lines or have a sequential number that can be used to uniquely name html form fields.

HTML example :

```
<body>
        <table>
                <!-- dl4c(for,sequence) -->
                <tr>
```

```
                    <td>dl4v(sequence)
                    <td>dl4a(Date)</td>
                    <td>dl4a(Invoice)</td>
                    <td>dl4a(Amount)</td>
            </tr>
            <!-- dl4c(next) -->
</table>
</body>
```

Another available syntax is dl4c(For,*fieldname=number*) , where *number* is the starting element # in the arrays to merge. This allows for ignoring beginning columns in the arrays, such as column headings.

URL encoding :
A field value may need to be merged into a web page using URL encoding. This functionality would be needed if merging in a variable link (ahref) which contacts value pairs which contain spaces. The URL encoding feature uses a function notation; the syntax "URL(value-name)" can be used in "dl4v" or "dl4a" references in place of a simple "value-name" (for example, "dl4v(url(value))"). The value of "value-name" will be encoded in URL format and substituted for the "dl4v" or "dl4a" reference.

dl4o(include, “filename”) Webserver side dynamic page includes
Allows the html page to include another html page. The filename can include merged values from the application server to create dynamic includes.
If the included filename is not in the current directory, dynamicXport will traverse its parent directory, up to the document root directory, to locate the included filename. The dl4o option appears in an HTML page within an HTML comment..
Here are two examples.

```
<!-- dl4o(include,"include_this_page.html") -->
<!-- dl4o(include,"dl4v(test)") -->
```

Automatic Hidden Values

The following will be automatically passed by DynamicXport server as hidden values under the following conditions :
Session will always be passed as the current session #.
Option will always be passed except to the Options Menu as the current Option ID.

Testing method

1. Create a text file of session variables and other variables needed for input into the html page. (A test results file from an application test can be used). Place the file in a read/write accessible directory on the web server and verify the file protection is read accessible. This simulates Session variables and inputs needed to display the HTML page.

2. Through the browser access the 'Test HTML Page' page from the DynamicXport menu and then proceed as follows :
Enter the HTML page filename to test .
Enter the test input text filename in the read/write directory that contains the test input variables.
Then click the GO button to submit.
(Files are assumed to be in the S_Root directory unless an absolute path (begins with / or \) is given.)

3. The page will post to the dxserver.cgi
Dwserver.cgi will recognize that you are posting from the test page and will read the values in the test input text file and Merge Output to the Output= html page. If the page is then submitted, the cgi will again recognize that you are in test mode, and instead of passing the information through the socket to the application server it will output the submitted fields to a text file in the same directory as the input text file with the same name as the input text file, with a out.txt extension. (NOTE: This will overwrite an existing file of the same name!)
(This file can be used as the input files for Application testing)
A 'Test Complete' page will then display.

The format of the test text file is as follows :
Fieldname=fieldvalue<cr>

Preceed list items with an L: or l:
Continue list with indented white space (spaces or tabs)
Preceed hidden items with an H: or h:

The text file can contain comment lines. Comment lines are any lines that begin with a semi-colon (;).

(Alternatively you can set the Output= variable in the text file and leave the HTML page prompt blank on the 'Test HTML Page'.)

Developing in FrontPage

Settings
DynamicXport is designed to be compatible with IIS or Apache Servers without FrontPage extensions and Explorer & Navigator browsers 4.0 and later.
You MUST go into Tools, Page Options and set parameters to limit options to be compatible with these restrictions, if you wish to also adhere to these restrictions.

On the menu bar click Tools, Page Options and then the Compatibility tab.
Set Browsers to 'Both Internet Explorer & Navigator'.
Set Browser versions to 'Custom'.
Set Servers to 'Apache server'.
Do NOT check the 'Enabled with FrontPage Server Extensions'.
Checkmark JavaScript, Dynamic HTML, Frames, CSS 1.0 (formatting)

Guidelines
FrontPage extensions do NOT need to be installed on the server.
Only those FrontPage components (webbots) that affect the development process are used by DynamicXport.

Standard Page Templates

Several general purpose page templates are provided in the dx/dxstd for your use. The HTML pages can be modified to create a different presentation.

validate.html
This page is automatically presented to the user when it is necessary for them to revalidate themselves. This occurs when the user's session lasts longer than the SYSTEM.Elapsed parameter or their session's inactivity is longer than the SYSTEM.Timeout parameter or if their IP changes during a session. The page requires that the user re-enter their User ID and Password to continue the session.

passwd.html
This template is available if you need a page to allow the User to change their password.

logoff.html
This page is presented when the user clicks on the logoff button. Logoff deletes the current session and then presents this page. You can change the logoff message and appearance to your liking. If you do not want a logoff page to display you can remove the comments (//) in the initial Javascript to go directly to the Exit URL.

msg.html
This page is available if you need a page to display various messages with one or more buttons. The same page can be used repeatedly for various messages because the contents of the page are dynamic from variables sent by the application.
The page accepts the following variables from the application :
dl4v(S_Msg) can be set to any text message you wish to display. Not required
.
dl4v(Msgtitle) can be set to any text page title you wish to display. Not required.

dl4v(Msgclass) indicates the class to use from the Stylesheet file when displaying the message. Not required. For example Msgclass=Error will display the text as defined by the Error definition in the StyleSheet file.

dl4v(Button) can be set to text of a button. Not required, if not set no button will display. If the contents of Button ends with .gif or .jpg the named image will display instead of a button.

dl4v(Next) can be set to indicate page to present when the button is clicked. If numeric, the browser will go back 'x' pages. If set to 'Option=*OPTIONID*' the browser will run the option as though selected from a menu.

If more than one button is desired then dl4l(Buttonlist) and dl4l(Nextlist) variables are used to define the buttons instead of dl4v(Button) and dl4v(Next).

Special fields :
dl4v(S_Msgno) can be used to display a message number or error number, but it is 0 by default and typically not displayed on the web page. If the S_Msgno variable is set by the application, DynamicXport will search the dxmsg.txt file (as determined by the User's View definition) and overwrite the S_Msg variable with the text corresponding to the message number.

dl4v(S_Intmsg) can be used to display an internal message but it is 0 by default and is intended to not display on the web page. The purpose of the S_Intmsg variable is so that the application can set an internal message, typically an error message, which will be recorded in the Audit log file.

S_Intmsg and S_Msgno are special session variables in that the application can set them and they are recorded in the Audit log file, even if Extended log is off.

dl4v(Auditlink) if Extended Log is on and the user has access to the DXAUDIT option this field is passed to provide a direct link to the Extended Log record contents. This field is populated by DynamicXport, not the application.

(The application must set Output=dxstd/msg.html or Output=*S_VIEW.OptionDIR*/msg.html to merge properly)

Look and feel of user's browser interface

Views and Option Types

VIEW Objects define the type of browser interface the user will experience. VIEW Objects define the method in which the User's Menu Options are displayed, the Images and Icons that will be used, the Stylesheet (defines fonts, colors) that will be used and the Directory to find the custom HTML files for the site.

The ORGANIZATION Object definition defines the VIEW Object that User's within the Organization will experience.

Each User is assigned to an ORGANIZATION Object.

Option Types

DynamicXport comes with several standard methods of displaying the user's menu options. The Option method that a user will receive is dependent on the VIEW object that is being used by their Organization. The Option type is defined in the VIEW Object.

Option Types are :

Type 1 Top frame with option dropdown box
Type 2 Left frame with options listed
Type 3 No frames/Full Page (typically large ICON images)
Type 4 Top frame horizontal tabs
Type X Custom user-defined file

Standard Option Types are pre-configured and simply need to be referenced by the VIEW Object.

The default set of HTML files for Options are located in the dx/dxstd directory.

If you wish to create a custom menu presentation, create a new html file called 'optionx.html' (or copy one of the standard option html files). Then create or change a VIEW object to link to the Custom Option by changing the Options Type code to type X-Custom.

To create a new Options look and feel you can create a new Options theme directory by copying the dx/dxstd directory to another directory under dx/ and then modify the options files.
For example copy dx/dxstd/* to dx/siteabc/*.
Create or change a VIEW object to link to the new Options directory by changing the Options Directory field.
The images directory and icons directory are subdirectories of the options/theme directory.

.css Stylesheet files

All DynamicXport Administration functions except User and Option maintenance use the default filename of 'dx/styles/dxstd.css' to locate Stylesheets.
Users and User's Options maintenance use the Stylesheet defined by the User's Organization's designated VIEW Object.
DynamicXport is designed so that custom applications use the Stylesheet defined by the User's Organization's designated VIEW Object.

Stylesheets

All Stylesheets are stored in the dx/styles directory.
The default Stylesheet file is dx/styles/dxstd.css.

To create a new Stylesheet copy the dxstd.css file to another filename under dx/styles and then modify the .css file. For example copy dx/styles/dxstd.css to dx/styles/siteabc.css
Create or change a VIEW object to link to the new .css file by changing the StyleSheet field.
(NOTE: Not all documented CSS1 functionality is properly supported by various browsers. For example margin and padding definitions in tables is not supported in Netscape 4 and should not be used. Take caution when adding definitions to the .css file and perform testing on browsers that will be supported)

Images and Icons files

The default set of image files and Option Icon files are stored in the dx/dxstd directory (except login page images).

Small and large Images for options, referred to as Icons, are stored in the subdirectory /icons.

You can add small and large Images for your options to the dx/'theme'/icons directory.

Other images related to Options are in the dx/'theme'/images directory.

Images not related to selecting Options should be stored elsewhere! (Typically within the directory structure of the HTML that will be using the images, which will be defined as the VIEW Merge Directory)

Images used for the login.html screens are located in the dx/login directory.
The default set is stored in the dxstd directory.
The reason login page images are kept separate is because the user is not yet identified until logged in so the system cannot determine the options set to use.

HTML Code Changes

In short, do NOT modify any HTML files that begin with 'dx'.
These would most likely be in the dx/admin directory.

Any HTML in dxstd directories can be modified, however they should be copied to another directory (theme) and modified there. These files will not begin with 'dx', indicating you can modify them, but you should put them in a theme other than dxstd first.
Before installing future releases you should backup entire dx/ directory.
Any files in cgi-bin, admin and dxstd directories may be overwritten with new versions and you may need to restore your modified versions or redo your modifications. Hence the advice to copy them to another directory.

Web Designer Tips

Javascript
See Appendix A for Javascript utilities available in the Includes directory.

Dynamic HTML with Javascript and merged indicator
There are many instances where you may want to display certain information or allow certain functionality on a web page depending on the privileges of the user, which is often determined in the application.
An easy method for accomplishing this is to set an indicator or flag in a dl4v field in the application.
The HTML page can then use Javascript to compare to the merged value to determine what to present on the web page. There are many examples of this methodology in the dx/admin html files of dynamicXport, in particular whether to display a More link on a list page to request more records.

Here is sample Javascript :

```
<script language="javascript">
  if ("dl4v(candoit)" == "yes"
          {
          document.write('You can see this');
          }
</script>
```

Scroll to anchor
The following Javascipt technique can be used to scroll to a particular anchor on a page after the page loads.

In dl4 set a field named "scrollto" to the name of the anchor to scroll to, for example :
Call DXSet("scrollto","thislocation")

In html :

```
<head>
<script language="javascript">
  function ScrollTo(theAnchor)
    {
    if (document.all)
      document.all[theAnchor].scrollIntoView(true);
    else
      window.scrollTo(0,document.anchors[theAnchor].y);
    }
</script>
</head>
<body onLoad="ScrollTo('dl4v(scrollto)');">
Page contents
<a name="thislocation">more page contents</a>
```

Unibasic/dL4 Developers Brief

The purpose of this brief is to provide instructions to develop custom Unibasic or dL4 code that would be integrated to an internet or intranet site using the DynamicXport platform.

A basic understanding of DynamicXport's role is necessary.

Application Interface Calls

Variable values are received and sent to HTML pages by using special DynamicXport Call commands. DXOpen receives all the variable values from forms fields submitted from the HTML page. DXGet is then used to read in the various field strings and string arrays. DXSet is used to set string and string array variable values for merging into an output HTML page. DXClose is used to indicate that all the output variables are set and the merge to the HTML page can begin. (The application can continue to do further processing after the DXClose is issued)

Call DXOpen() **Unibasic syntax Call $DXOpen**

Used by applications to read a value pair string from the input pipe (standard input). The values in the value pair string are loaded into the input table for later access by DXGet().

Call DXClose() **Unibasic syntax Call $DXClose**

Used by applications to write all values in the output table to the output pipe (standard output) as a single value pair string.

Call DXGet(Name$,Value$) **Unibasic syntax Call $DXGet,Name$,Value$**
Call DXGet(Name$,Value$[]) **(dL4 only)**

Used to retrieve the value of the field Name$ into Value$. If the value is a list and Value$ (not Value$[]) is used, then each list value will be stored sequentially into Value$ with a string terminator (a binary zero character) between each value and a double terminator at the end of the list. If Value$[] (a string array) is used, the list values will be assigned to successive elements of the array (Value$[0], Value$[1], . . ., Value$[N - 1] where N is the number of values in the list). Value$[N], if it exists, will be assigned "". If Name$ does not exist in the input table, a single "" value will be returned. Fieldnames are case insensitive.

Example :
HTML :

```
<HTML>
<BODY>
<FORM ACTION="../cgi-bin/dxserver.cgi">
<INPUT TYPE="text" NAME="CustID" SIZE="10">
<INPUT TYPE="submit" NAME="Submit">
</FORM>
</BODY>
</HTML>
```

Unibasic :

```
100 Dim C$[10]
200 Call DXOpen()                    !Brings web form inputs into application for DXGet access
300 Call DXGet("CustID",C$)          !Looks for field called CustID and puts value in C$
```

Call DXSet(Name$,Value${,Type$}) **Unibasic syntax Call $DXSet,Name$,Value${,Type$}**
Call DXSet(Name$,Value$[]{,Type$}) **(dL4 only)**

Used to set the value of the field Name$ to Value$ with type Type$. Type$ must be "v" (value), "l" (list), or "h" (hidden). Type$ is case insensitive and so "V", "L" and "H" are valid type values. If Type$ is not specified, the type is assumed to be "v" if Value$ is specified or "l" if Value$[] is specified. If Type$ is "l" (list) and Value$ is passed, then Value$ must be a concatenated list of values with each value separated by a string terminator (binary zero character) and double terminator marking the end of list. Null ("") values in the list must be represented by at least one space (" "). If the list is passed as Value$[] (a string array), each value must be stored in a separate array element starting at Value$[0] and ending at the end of the array or the first "" value.
You can perform a DXSet using subscripts of a string variable, for example DXSet("Name",A$[20,46]). Note that field names are case insensitive.

Example :
Unibasic :

```
100 Dim C$[10]
200 C$="123456"
300 Call DXSet("CustID",C$)          !Sets the value of field called 'CustID' to contents of C$
400 Call DXClose()                   !Sends field names and values set by DXSet to be merged by
                                      DynamicXport with the HTML page
```

HTML :

```
<HTML>
<BODY>
The customer's ID is dl4v(CustID)
</BODY>
</HTML>
```

Methods of Maintaining State

1) The best method of maintaining state of an ongoing session is to use DynamicXport's U_ fields.
A field starting with U_ indicates a Session User-defined field.
If you want to maintain a variable from one HTML page to another, such as an Order number, you can set a variable U_Order = 12345 in your application using the Call DXSET command.
DynamicXport will retain this value for the life of the user's session. Even if the user leaves your site and comes back, as long as the session has not timed out the U_ variables will be stored in the Session's record. Your application would need to use the DXGET command to get the value of the variable.

2) You can pass hidden values from page to page. The only weakness is if the chain is broken (ie user leaves site and comes back) the values are lost.

Reserved FORM fieldnames

You can merge any number of string variables or lists of string variables from your application to an HTML page. There are several reserved fieldnames that are used by DynamicXport and cannot be used for application specific fieldnames. Some can be read by the application. Some can be set by the application with knowledge of it's functionality.

"Option"	
"Action"	value can be set by the application
"Session"	
"E_xxxxxxxx"	
"S_xxxxxxxx"	
"ID"	
"Password"	available only when application is accessed by ID & Password and is encrypted
"Output"	value can be set by the application
"HTML"	value can be set by the application
"XML"	
"SQL"	

These fieldnames are used as follows :

Option

Must allows be present. Indicates which 'menu' option (program) to process.
Reserved global options are :

- Option=Login (to login, requires ID and Password fields also)
- Option=Validate (to revalidate a user if re-authentication needed)
- Option=Options (to display menu options)
- Option=Logoff (logs off user and ends user session)
- Option=Revision (returns revision # of DynamicXport from Application Server)

Session

The current session ID. Either the Session ID or the User's ID and Password must be present with Option in order to process. Options may be designated to only run if the session ID is provided or they can be configured to run if session or ID and password is provided.

E_xxxxxxxx

All fields prefixed with E_ are reserved by DynamicXport to pass standard CGI environment variables. Some environment variables may not be available depending on the web server or the browser.

S_xxxxxxxx

All fields prefixed with S_ are considered DynamicXport session parameters. HTML pages can access these parameters by name.
(Also custom fields can be stored with the session by the application. These fields begin with U_ to
indicate User Defined session field.)

ID
Is the User's ID. Always used in conjunction with password.

Password
Is the User's password. Always used in conjunction with ID.

Action
An action to be processed in relation to the Option indicated. This field is optional.
Reserved global actions are :
All Global Options are also reserved and available as Global Actions
Action=Input (to display the Input HTML document for the option first. This is typically set when running an option from a menu)
Action=Sort
The Action fieldname can also be used within applications. Some suggested Action keywords are 'HELP','SEARCH','ADD', 'SAVE","CHANGE', 'DELETE' ,'LIST', etc.

Output
Output can be used to override the Output Merge filename if it varies depending on option results (ie. Multiple input pages, such as order entry, all using one option, with a parameter indicating the page# being submitted)
NOTE: If the Output Merge filename is overridden by setting the Output fieldname in the application, the field will automatically be preceeded by the S_ROOT value, but not the S_MERGEDIR value.

HTML
This field name is used by applications to output HTML code back to the client. In the application the developer can set HTML= any html code. Then this HTML will be merged into the Output HTML file where the HTML variable is designated. This field provides the same functionality as the Print statement when using the CGI Library.

XML Reserved

SQL Reserved

Reserved Environment Variables

On Web Server
(these variables are defined in the dxserver.txt file)

System Variables
S_ROOT Used to define the root directory on the Web Server to locate the dx/ directory.
(Known roots : Apache ../ NT IIS /dx/)
On installation DyamicXport will configure this automatically. It only needs to be defined in the dxserver.txt file if the default needs overridden.

S_CGI Path to find the correct CGI to process requests. Dxserver.cgi and dxserver.txt pairs can

be copied and modified to contain different parameters on a single web site. Multiple login pages can then be created to post to the different dxserver.cgi's. This method allows a single web site to 'talk' to multiple independent application servers using multiple dxserver.txt configuration files.

S_URL The full URL (ie http://www.dynamic.com) to the site where the dx/ directory resides.

S_SecureURL The full URL (ie https://www.dynamic.com) to the secure site where the dx/ directory resides.

S_Cookie If set to yes, the session ID will be stored on the client machine as a cookie. The option page will validate the session by comparing to the cookie. (not implemented)

S_ContentOut If not present defaults to DXCONTENT. If present, specifies type of content to send out.

DXIP Used to designate the IP address of the DynamicListener on the Web Server

DXPORT Used to designate the Port # of the DynamicListener on the Web Server

DXCONTENT Used to designate format type of input/output, ie HTML,XML.
Default is 'HTML' at installation.

DXREADTIMEOUT Used to designate time in seconds to wait for a response from the socket.
Default is '180' seconds at installation.

DXBUFSIZE Default is 128000. The number of characters (16-bit Unicode character set) to send or receive through the socket in any one transmission. Minimum is 1024 characters. Decreasing can enhance performance, increasing can reduce performance. Should match BufSize in dxsystem.txt file.

ConnectRetryCount Designates number of retries to open the socket before failing
Default is 3.

CheckBrowser=WML If present, web server checks if user's browser supports WML (wireless markup language) and if so overrides S_ContentOut with WML.

E_*environmentvariablename*
Listing any standard web environment variable names proceeded with an E_ will make that environment variable available to the application programs.

On Application Server
(these variables are used internally and do not need to be configured)
DXTESTIN Used internally to designate name of the 'test input file' to read when running an application test.

DXTESTOUT Used internally to designate name of the 'test output file'.

Limitations

For User defined session variables (U_), the fieldname is limited to 32 characters and the data field is limited to 254 characters. The data storage limit of all combined U_ fields is 4000 characters.

Messages stored in the Application Messages file and referred to by message number are limited to 1900 characters.

A dl4v merge into html is limited to 1000 characters per html line. To merge larger strings use dl4t in the html code.

If your application server is to experience large dynamicXport user volume, a hardware passport is not appropriate because of it's current speed limitations to verify valid licensing.

Standard Output Templates

See Web Developers Brief/Standard Page Templates for definitions of standard pages you can merge to and fields to set.

Testing method standalone

1. Create a 'test' text file to simulate initial inputs needed for the application coming from the web page using the format described below. The filename can be anything you choose.
(Recommended that the name of the test file be related to the particular application you are testing, so you can manage different test files. For example 'testarinquiry'.) The file needs to contain any input variables needed by the application.

2. Run dxtest.dl4 or dxtest.ub or create your own program to set the ENV variable DXTESTIN to the input filename to read on a DXOPEN command and DXTESTOUT to the output filename to write results to on a DXCLOSE command.

a) The screen will display the current settings of the ENV variables
   DXTESTIN is set to
   DXTESTOUT is set to
b) You will be prompted to enter a new input text filename. You can enter nothing to leave the filename set as is.
c) You will be prompted to enter a new output test filename. You can enter nothing to leave the filename set as is.valid
d) The new environment variable settings will then be displayed.

3. You can now run your application. DXOPEN will get it's inputs from the filename defined in DXTESTIN and write it's output to the filename defined in DXTESTOUT. Note, the filename DXTESTOUT will be overwritten on each DXCLOSE process.

The format of the test text file is as follows :
Fieldname=fieldvalue<cr>

Preceed list items with an L: or l:
Continue list with indented white space (spaces or tabs)
Preceed hidden items with an H: or h:

The text file can contain comment lines. Comment lines are any lines that begin with a semi-colon (;).

Development Tips

Record Reads
It is important to develop your code to deal with locked records. Use ROPEN statements to open and read files you are not updated. Or use the timeout parameter on your Read statements.

Record Locking

Since the CGI Socket does not remain open between web pages, if you will not be able to 'lock' records during maintenance. There is a suggested method for verifying a record that is being updated has not been modified between pages to simulate the record locking function by using the CRC32 intrinsic function in dL4 or Unibasic. A dL4 example :
CRC32$ = Str$(CRC32(Hex$(Org.))) where Org. is the full record structure of the record to be modified
Call DXSET("CRC",CRC32$, "H") this will merge the original checksum into the form page
When the form is submitted, the application would first read the record to be updated, recalculate the checksum and compare that the checksum has not changed.

Error trapping
You should develop your application program with some form of error trapping such as an If err 0 routine or Try/End Try routine to prevent the application from simply causing a time out if a program error occurs. See the program examples in Appendix D (excust example starting at line 4000). It is also recommended that you take advantage of the standard msg.html page and standard S_Msg, S_Msgno and S_Intmsg variables.

Standard MSG page
Refer to the documentation of the msg.html page in the Web Designers Brief section for an explanation of the fields that can be merged to a standard message page and the special S_Msgno and S_Intmsg fields.

Continued Processing
Your application is considered complete by DynamicXport when a DXCLOSE statement is processed. At that time all the set fields are sent through the socket to be merged with the output page. Your application program can continue to do additional processing such as file updates and internal printing after the DXCLOSE statement. Any processing done after the DXCLOSE statement will not affect the response time to the user. Processing after the DXCLOSE statement CAN NOT use STDIN or STDOUT, thus INPUT statements or screen PRINT statements can not be processed.

Table Column fields
A table merge is terminated on the first occurrence of a column with a null value.
If no table rows show on the web page verify all the column fieldnames are being populated. If one of the columns is not being populated it will cause the first row of that column to be null which will terminate the merging of the entire table.
If the table appears incomplete it indicates that a column may have been populated inadvertently with a null which causes the table merge to terminate. Be sure if you have potential null fields in a record that you change them to a space character in the string array to avoid this error.

Passing DXOpen parameters
Once you have performed a DXOPEN statement the input fields sent by the web page are in memory and no longer in the input pipe so any future DXOPEN statements would result in losing the sent information. To pass the information received by the DXOPEN statement and subsequent DXGETs to another program for further processing, you can Call the next program by program name (Unibasic style Call) (as opposed to CHAIN or SWAP) and then have an Enter statement in the called program. If the called program needs to be accessed as standalone or callable you can use the following statement in dL4, or similar in Unibasic :

Try Enter A$,B$ else Call DXOpen()

If the routine is called, previously read DXGet values can be passed a Call parameter or you can execute DXGets and DXSets in the called program.
A chained to or swapped to program will NOT be able to perform DXGets or DXSets as it is a separate process and would not have access to the DXOPEN and DXCLOSE tables.

Unibasic List Variables
Since Unibasic does not have string array capability, to get or set list types requires a single string with a string terminator between each value.

Use the following technique to get a list :
```
100 Call $DXGET,"fieldname",B$
200 L1=1
300 A$=B$[L1]
400 L1=L1+Len(A$)+1
500 IF A$="" GOTO end
600 process A$
700 Goto 300 !get next item in list
```

Use the following technique to set a list :
```
100 L1=1
200 set A$
300 IF A$="" GOTO 700
400 B$[L1]=A$
500 L1=L1+Len(A$)+1
600 Goto 200 !set next item in list
700 B$[L1]="" !set final null
800 Call $DXSET,"fieldname",B$
```

LUMAP or LUST setting
Your application programs and files typically will not reside within the DynamicXport directory structure. Thus, the LUMAP environment variable and possibly others need to be set prior to DynamicXport launching a dL4 application program. The LUST environment variable and possibly others need to be set prior to DynamicXport launching a Unibasic application program. Environment variables are set in the Organization's Prelink field or overridden per Option in the Option's Prelink field.
An example dL4 Option configuration is :
Option Prelink : LUMAP="1/=/tmp"
Option Link : 1/example.dl4
The program and related files would reside in directory /tmp

An example Unibasic Option configuration is :
Option Prelink : LUST=/usr/lib/ub:/tmp
Option Link : 1/example.dl4
The program and related files would reside in directory /tmp, the /usr/lib/ub is needed to find Unibasic related files.
(If accessing Universal files in Unibasic, ISAMSECT must be set equal to 8 or lower.)

The rule for formatting the Prelink string is as follows:
Each environment variable is a value/pair list. The left side of the equal sign is the environment variable name. The interesting thing is the separator character. By default, a space is the separator character. However, if the first chracter in the value is " or ', then the value must terminate with the same character.
For example, LUMAP="example=Program Files\dl4\files" LIBSTRING='d:\Program Files\lib'

URL link types
For URL link types, if the Input Page field is used a merge output will be done to the page (the page must be in the Merge Directory) vs. if the Link field is used a redirect to the page will occur. The redirect will assume the page is in the Merge Directory unless a full URL is given.

Caution using U_ fields for checkbox values
You can use U_ fields to retain the value of the field for the duration of the session. Checkboxes by nature will send a value through the socket **only** if the box is checked. If the box is not checked no value will be sent by the browser. If the checkbox value was stored in a U_field, the U_field value would remain as if

the checkbox is checked. To avoid unintentional results it is recommended to not use U_fields in conjunction with checkboxes.

Use of 'FX' and 'FM' mnemonics in a merge field
By default DXSET fields are merged into the HTML document in a plain text format so that any characters merged are not interpreted as HTML directives.
If the 'FX' mnemonic is embedded in a field, plain text format is turned off during the merge of the remaining data of that field value (or list element) or until an 'FM' mnemonic is encountered.
If you wish to merge in HTML directives you will need to proceed them with an 'FX' mnemonic.
Use of 'SI' mnemonic in a list
When creating a string list array, a particular item or items can be designated as selected when merged by embedding the 'SI' (or 'SELECT' in dL4) mnemonic at the beginning of the field.
Another method of setting the default selected item in a list is to use the javascript function DX_setselect (in includes/dxutil3.js) when the html page loads in the browser (<body onload="">).

SPC(6)
There will be a unique SPC(6) value assigned to each socket access through dynamicXport. Note that the SPC(6) value is only persistent to the extent of a single socket access. Another access through the socket by the same user (same session) could be assigned a different SPC(6) value than the previous access. The SPC(6) value can be up to 4 digits in length. You can use the SPC(6) value to create a unique filename for sorting, storing data, etc. If used in a filename it is recommended that STR(SPC(6)) or STR$(SPC(6)) to convert to string without leading or trailing spaces.

Security Precautions within the application
Do not issue a SYSTEM command using field values submitted by the user without first verifying the field's value is limited to acceptable and expected commands. Otherwise, a user could send a system damaging command in a browser field.

URL encoding
Unfortunately sometimes the information you dynamically merge into a web page needs to be URL encoded. If a field value could potentially contain spaces or other URL encoded characters and the field will be used in an <a href> link or javascript href, the field will need to be URL encoded to function properly. The easiest way to accomplish this is to URL encode a field when it is merged using the syntax of dl4v(url(value)).
To URL encode a variable within a dL4 application, use the new intrinsic CALL, Call DXURL(destination$, source$) or the new intrinsic function DXURL$(source$). In Unibasic use the new intrinsic CALL, Call $DXURL,destination$,source$.

Clearing U_fields
All U_fields can be cleared for the session by setting a special ‘S_Clear’ fieldname to a ‘Y’.
i.e. Call DXSet(“S_Clear”,”Y”) !Clear all U_fields

DXBridge Access

A program, **dxbridge.dl4**, provides a method for a character based Unibasic or dL4 application program to talk to dynamicXport. Within the application the dxbridge.dl4 program can be accessed with a Call statement in dL4 or with a System command in dL4 or Unibasic. The simplest method is with a Call statement in dL4.

When accessed with a System command dxbridge.dl4 accepts two mandatory arguments, the name of a text file containing the input value/pair fields the dynamicXport application is expecting and the name of a text file for dynamicXport to save the result value/pair fields to. This can be viewed similarly to how a developer tests an application using the standalone method.

An optional argument specifies the dynamicXport listener's IP address:port number. If it is omitted, then dxbridge.dl4 will use 127.0.0.1:9632 as a default value.

Note: In order to access dynamicXport options via the bridge, the IP of the server running the dxbridge.dl4 program must be listed in the dxsystem.txt file as a BRIDGEIP, i.e. BRIDGE1IP=127.0.0.1
The IP must also be listed as a WebServerIP.

Note: To access a High or Medium security dynamicXport option the User ID and Password who has access to the option must be included in the input value/pair list file. It is recommended that High or Medium security options only be accessed through the bridge on the application server itself (the local IP), behind a firewall.

The following example shows a dL4 program which uses a Call statement to change a User's e-mail address through the bridge.

```
Declare Intrinsic Sub DXGET
Declare Intrinsic Sub DXSet
Declare Intrinsic Sub DXCopy

!Create a value/pair field list
Dim A$[8],B$[20],C$[8],Status$[72],email$[50]
Input "Enter your User ID : "A$
Print
Input "Enter  your Password : "B$
Print
Input "Enter User ID you wish to change : "C$
Print
Call DXSet("ID",A$) !set login User ID to access dynamicXport option
Call DXSet("Password",B$) !set password to access dynamicXport option
Call DXSet("Option","DXUSER") !access this dynamicXport option
Call DXSet("Action","Change") !action to read a record to modify
Call DXSet("UserID",C$) !read this user's record
!Call dxbridge.dl4 program
Call "/usr/xport/dxbridge.dl4"
!The above will cause the dxbridge.dl4 program to connect to dynamicXport on 127.0.0.1:9632
!dynamicXport will process the Option request
Call DXGET("S_MsgNo",Status$) !Check the error status
If Status$
  Then
   Print "Error: ";Status$
Else
  !a special DX call DXCopy copies all value pairs from output as though a DXSET was
  !done for each pair. This is helpful if not all the fields are known.
  Call DXCopy()
  Call DXGET("UserEmail",email$) !Get current e-mail for display
  Print "The current email address is ";email$
  Input "Enter new e-mail address : "email$
  Print
  Call DXSet("UserEmail",email$) !Set the new value
  Call DXSet("Action","Save") !action to save a record
  !Call dxbridge.dl4 program with input file and output file arguments to save
  Call "/usr/xport/dxbridge.dl4"
  Call DXGET("S_MsgNo",Status$) !Check the error status
  If Status$
   Then
    Print "Error: ";Status$
```

```
  Else
    Print "Done"
  End If
End If
```

The following example shows a dL4 program which uses a System command to change a User's e-mail address through the bridge.
Similar code would be used in Unibasic to access the bridge with a System command.

```
Declare Intrinsic Sub DXOpen
Declare Intrinsic Sub DXGet
Declare Intrinsic Sub DXSet
Declare Intrinsic Sub DXClose
Declare Intrinsic Sub DXCopy

!Create a value/pair field list to save in an input text file
!This can be done by simply writing to and closing a text file also
Dim A$[8],B$[20],C$[8] ,Status$[72],email$[50]
Dim Inputs$[30],Outputs$[30],S$[100]
Input "Enter your User ID : "A$
Print
Input "Enter  your Password : "B$
Print
Input "Enter User ID you wish to change : "C$
Print
Call DXSet("ID",A$)                    !set login User ID to access dynamicXport option
Call DXSet("Password",B$)              !set password to access dynamicXport option
Call DXSet("Option","DXUSER")          !access this dynamicXport option
Call DXSet("Action","Change")          !action to read a record to modify
Call DXSet("UserID",C$)                !read this user's record

!save the value/pair field list to a text file, using a variation of DXClose, specifying the filename
Inputs$="input",STR$(SPC(6)),"!"       !make it a unique filename
Outputs$="output",STR$(SPC(6)),"!"
Call DXClose(Inputs$)

!Call dxbridge.dl4 program with input file and output file arguments
S$="/usr/bin/run /usr/xport/dxbridge.dl4 " + Inputs$ + Outputs$
System(S$)
!The above will cause the dxbridge.dl4 program to connect to dynamicXport on 127.0.0.1:9632
!dynamicXport will process the Option request and output the results in the outputfile

!read the value/pair field list output file, using a variation of DXOpen, specifying the filename
Call DXOpen(Outputs$)
Call DXGet("S_MsgNo",Status$)          !Check the error status
If Status$
 Then
   Print "Error: ";Status$
Else
 !a special DX call DXCopy copies all DXOpen value pairs for output as though a DXSet was
 !done for each pair. This is helpful if not all the fields are known.
 Call DXCopy()

 Call DXGet("UserEmail",email$)        !Get current e-mail for display
 Print "The current email address is ";email$
```

```
 Input "Enter new e-mail address : "email$
 Print
 Call DXSet("UserEmail",email$)              !Set the new value
 Call DXSet("Action","Save")                 !action to save a record
 Call DXClose(Inputs$)              !save text file to write back all values

 !Call dxbridge.dl4 program with input file and output file arguments to save
 System(S$)
 Call DXOpen(Outputs$)
 Call DXGet("S_MsgNo",Status$)               !Check the error status
 If Status$
  Then
    Print "Error: ";Status$
 Else
    Print "Done"
 End If
End If
```

DXPurge

A program, **dxpurge.dl4**, provides a utility for purging old session and audit records from dynamicXport. The program is automatically started the first time the application server is accessed.
This process runs continuously, waking up periodically to determine if there are dead sessions or old audit records to purge.

The program uses the parameter PurgeAuditAge in the dxsystem.txt file to determine the number of days to leave in the audit file and the parameter CronFrequency in the dxsystem.txt file to determine how often (frequency) to wake up and check for records to purge.

Uploading Files

dynamicXport users may upload both text and raw files from their local system directly to the application server. These files are immediately made available to application programs for any custom processing. Uploaded text files may be tranformed from one format to another, e.g. DOS to UNIX, UNIX to DOS, etc. The following configuration options in the dxsystem.txt file provide file upload capability and limit the number and size of files that can be uploaded.

UPLOADMAXFILESIZE = the maximum file size in bytes that may be uploaded
UPLOADMAXFILES = the total number of files that may be uploaded at once

An html form page that uploads a file must have enctype="multipart/form-data" in its form tag and the filename to be uploaded must be entered with a 'file' input type.

The file upload feature is enabled by setting the S_UPLOADFILE file option in the dxsystem.txt file. Uploaded files are mapped according to rules specified in the dxsystem.txt file.
The S_UPLOADFILE option in the dxsystem.txt provides a global mapping option.
In addition, S_UPLOADFILE may be set as part of a session to override the global setting. The following syntax describes the S_UPLOADFILE option.

S_UPLOADFILE = [(option)] [<protection>] "filename_macro[!]" AS "destination_file_type"

option := [CASE = U | L | A]
[, ACCEPT_EXT = ALL | space separated list of extension, e.g. .gif, .txt]
[, APPEND = Y | N]
[, EXTENSION = add this extension if the source file lacks this extension]

CASE = U convert destination filename to uppercase
CASE = L convert destination filename to lowercase
CASE = A accept destination filename as is

ACCEPT_EXT = ALL allow files with any extension to be uploaded
ACCEPT_EXT = .gif .exe .txt
a list of space separated file extensions that may be uploaded. Files without these extensions may not be uploaded.
Please note that file uploading capability is disabled if ACCEPT_EXT is undefined.

APPEND = Y append to destination file
APPEND = N do not append to destination file

EXTENSION = xxx add the extension, xxx to the destination filename if the destination filename lacks such an extension

<protection> destination file protection, e.g. <666>

filename_macro := must begin with an absolute path and may contain
one or more of the following

%NAME%	use source filename
%MMDDYY%	current date
%MMDDYYYY%	current date
%DDMMYY%	current date
%DDMMYYYY%	current date
%YYDDMM%	current date
%YYYYDDMM%	current date
%YYMMDD%	current date
%YYYYMMDD%	current date
%DDDYY%	current date
%DDDYYYY%	current date
%TIME24%	current time (24 hour format)
%TIME12%	current time (12 hour format)
%TIMEPM%	current time (am or pm)
%FIELDNAME%	dynamicXport field name
%%	percent sign (%)
%# ... #%	use a sequence number using # as a mask value
%& ... &%	use a sequence number using & as a mask value

destination_file_type := RAW | TEXT | DOS TEXT | ANSI TEXT |
UNIBASIC TEXT | UNIX TEXT | MACINTOSH TEXT

The S_UPLOADFILE global setting may be overwritten by setting the S_FILEUPLOAD_form_field_name field prior to uploading.

After uploading, the S_FILEUPLOAD_form_field_name contains the name of the destination filename that was uploaded, and can be read by the application with a Call DXGet statement. For example, if the form field name is 'excelfile', then Call DXGet("S_FILEUPLOAD_excelfile",filename$) would return the full path filename of the file uploaded for that field.

An optional UPLOADTEMPDIR parameter can be defined in dxsystem.txt to define where temporary files are created during file upload. If UPLOADTEMPDIR is not defined (the default), the temporary files are created in the application directory (typically /home/dxport or C:\dxapp).

An example of an upload application can be found on the web server in examples/upload.html and on the application server in examples/exupload.dl4. Yhe example option is EXUPLOAD.
The UPLOADMAXFILESIZE, UPLOADMAXFILES and S_FILEUPLOAD fields may need to be set in dxsystem.txt for the example to function.

Example dxsystem.txt field definitions :
uploadmaxfilesize=100000
uploadmaxfiles=1
s_uploadfile=(ACCEPT_EXT = ALL) "/home/dxport/uploads/%NAME%" As "TEXT"

Downloading Files

dynamicXport users may download files directly from the application server to their client system. Both binary and text files may be downloaded. Only a single file at a time may be downloaded.

A file download HTML page must contain S_DOWNLOAD=1 (actually any value other than null) in its parameter list.

The S_CONTENTTYPE and S_CONTENTDISPOSITION settings in the dxsystem.txt file on the application server controls file download.

If not defined in dxsystem.txt, the default values are:
Content-Type: application/octet-stream
Content-Disposition: attachment; filename=filename_to_download

An application program must provide the name of the file to download, including path, by setting the S_DOWNLOAD name/value pair variable with a DXSet statement.
If an application program includes a S_DOWNLOAD value, then S_CONTENTDISPOSITION is sent to the browser, along with the file to download. Othewise, S_CONTENTTYPE is sent.

An example of a download application can be found on the web server in examples/download.html and on the application server in examples/exdown.dl4. The example option is EXDOWN.

Copying Files from Application Server to Web Server

An application can copy files from the application server to the web server by setting the S_COPYTOWEBSERVER value pair list variable. One typical use of this feature would be to allow selected users to update a web site by uploading HTML files which are then copied to the web server. Security is maintained because file copying is controlled by an application program and, as with any dynamicXport application, a user cannot invoke or even see an application unless the administrator has enabled that application for that user. Another use would be to publish files from the application server to the web server, such as general use pdf or html pages.

Each file to be copied is specified as a set of three consecutive values in the S_COPYTOWEBSERVER list. The first value of a triplet is the path of the file to be copied, the second value is the destination path of the file on the web server, and the third value is the file type ("Raw" or "Text"). The destination path must begin with a logical directory name ("directory/filename"). The logical directory must be defined in COPYMAP in dxserver.txt (see below). The destination path can be prefixed with a permissions option using the format "<xxxx>" where "xxxx" is a Unix style octal permissions value (such as "600") or any dL4 permissions value acceptable on the web server. If the file is to copied over any existing file, a trailing exclamation mark should be added to the destination path ("directory/filename!"). If the file type is specified as "Raw", the file will copied to the web server without any changes or conversions. If a file is specified as "Text", the file will be read as a text file and converted on the web server to the text file format needed by the web server.

The following example shows how a dL4 application can copy the files "example1.html" and "example2.html" to the web server by setting S_COPYTOWEBSERVER:

```
Dim Files$[6,64]
Files$[0] = "/home/dxport/newfiles/example1.html"
Files$[1] = "html/example1.html!"
Files$[2] = "Text"
Files$[3] = "/home/dxport/newfiles/example2.html"
Files$[4] = "html/example2.html!"
Files$[5] = "Text"
Call DXSet("S_COPYTOWEBSERVER",Files$[],"L")
```

Additional files could be copied by making the string array larger and setting additional source, destination, and type values. Please note that the actual copying of the files does not occur until after the application calls DXCLOSE.

The destination directory "html" used in the destination path in the example is not an actual directory on the web server. Instead, it is a logical directory defined in the COPYMAP list value in the dxserver.txt configuration file on the web server. For the example above, the dxserver.txt file might contain the following definition to define the logical directories "html" and "data":

```
L:COPYMAP=html=/usr/internet/ns_httpd/httpd-80/htdocs/htdocs
   data=/usr/internet/ns_httpd/httpd-80/htdocs/data
```

This translation of logical to actual directories makes it unnecessary to update applications whenever the web server directories are changed or if the web server is completed replaced. Use of COPYMAP is not only recommended but REQUIRED. Each line of the COPYMAP definition in the dxserver.txt file has the following form:

"logical-directory=actual-directory-path"

Each directory definition can include a permissions option "<xxxx>" to specify default permissions. If the permissions option is not used, the copied file will use the system default permissions. Copying files to the web server may fail due to incorrect file paths or permissions. Since the copying occurs after the application calls DXCLOSE, the application has no way to determine if the copy succeeded or not. Errors can, however, be reported to the user by displaying the S_COPYERROR and S_COPYERRORNAMES values in the output page. The S_COPYERROR value is the number of files that were NOT copied to the web server. The S_COPYERRORNAMES value is a list of the destination filenames of the file that were not copied. These values can be displayed in an output HTML page using the normal "dl4v()", "dl4t()", and "dl4l()" merge functions.

Installation

Installer/Integrator Brief

The purpose of this brief is to summarize the installation process and integration test process.
Please refer to the readme.txt file included in the installation download for the latest installation process.

On Windows Application Servers

Install dL4
See dL4 readme.txt for install instructions
License
See Passport readme.txt for install instructions
Install DynamicXport Components
Download dynamicXport for platform 1B and see dynamicXport readme.txt for install instructions

On Unix/Linux Application Servers

Install dL4
See dL4 readme.txt for install instructions
License
See Passport readme.txt for install instructions
Install DynamicXport Components
Download dynamicXport for the appropriate platform and see dynamicXport readme.txt for install Instructions

On Web Servers

Install DynamicXport Components
If same platform as the Application Server, copy the WebServer directory to the appropriate web server directory.
If a different platform, download DynamicXport for the appropriate platform and then copy the WebServer directory to the appropriate web server directory.

Licensing

Development Kit
Can develop and demonstrate applications with this license. Can add more Options at anytime. Restricted in that you cannot deploy a live commercial application.

Run-time or Application Server License
Enables the commercial deployment of web applications, from a single application server, developed using the Developer Kit. No development is available with this license. But adding additional Options is disabled.

Full Developer License
Full license allowing development and adding of Options and deployment of live commercial applications.

DynamicXport Lite License
When operating under a user-limited dynamicXport Lite license you will not be able to add more users than is permitted by the terms of the license.
Also each User ID can be logged into dynamicXport multiple times using a single internal IP address (i.e. a single intranet PC). DynamicXport Lite will not permit a User ID to be used by more that one IP address at

any given time. If a login using an active User ID from a different IP is attempted the user will be prompted whether they want 'Take Control' of the already active session or 'Cancel' the login attempt.

Configuring Socket Communications

Inetd on Unix Application Server

See readme.txt for example inetd install instructions
It is also recommended to place the Web Server IP address in the /etc/hosts/ file to improve Inetd's reverse DNS lookup performance.

CGI on Web Server

In the cgi-bin directory is a pair of files dxserver.cgi and dxserver.txt. Configuration parameters for the .cgi program are stored in the .txt file of the same name. Multiple pairs of these files can be created and stored in the cgi-bin directory if the same web server is used to communicate to multiple application servers. A separate login page should be created for each application server, posting to the appropriate .cgi program.

NT IIS configuration of CGI

Associating .cgi extension with dL4

On IIS you need to associate or map the .cgi extension to run the dxserver.dl4 program. Refer to the readme.txt install instructions for step-by-step installation.
This will map ALL .cgi's to the dl4 program! If you have other .cgi's existing on the site, you will need to 1)change the extension of dxserver.cgi to something else and 2)change the App Mapping in IIS to the new extension.

Configuration parameters are stored in a text file in the cgi-bin directory called dxserver.txt

S_ROOT	should be set to /dx/ On Windows XP try ../../ On W2K personal web server try ../../dx/

SCO Unix/Apache configuration of CGI

Configuration parameters are stored in a text file in the cgi-bin directory called dxserver.txt

S_ROOT	should be set to ../

DXSERVER.TXT Parameters

Definition of all configuration parameters stored in dxserver.txt in the cgi-bin directory :

S_ROOT	should be set to /dx/ for NT ../ for SCO Apache
S_URL	Set to the full URL (ie http://www.dynamic.com) to the site where the dx/ directory resides.
S_SecureURL	Set to the full URL (ie https://www.dynamic.com) to the secure site where the dx/ directory resides.
S_ContentOut	if not present defaults to DXCONTENT. If present, specifies type of content to send out.
DXIP	should be set to IP of DynamicListener
DXPORT	should be set to port of DynamicListener
DXREADTIMEOUT	default is 180.
CONNECTRETRYCOUNT	default is 3.
DXCONTENT	should be set to 'HTML'
DXBUFSIZE	default is 128000. The number of characters (16-bit Unicode character set) to send or receive through the socket in any one

transmission. Minimum is 1024 characters. Decreasing can enhance performance, increasing can reduce performance. Should match BufSize in dxsystem.txt file.

SHOWOUTPUTFILENAME default is N for No. If set to Y, if the web server cannot open an html output file, the missing filename is displayed in the error message. This can be useful if trying to determine missing filenames.

CheckBrowser=WML if present, server checks if user's browser supports WML (wireless markup language) and if so overrides S_ContentOut with WML.

E_*environmentvariablename*= if present, will make the web server environment variable available to the application programs. For example, E_HTTP_ACCESS= in the dxserver.txt file allows the application programs to access the HTTP_ACCESS environment variable as seen by the web server by getting field name E_HTTP_ACCESS.

Socket test

To do a basic test to verify the socket is properly functioning between the web server and the application server you can use the special Option=Revision form field which will return the DynamicXport revision number if the socket is functioning.
From the browser use the following address :
Domain name/dx/cgi-bin/dxserver.cgi?Option=Revision

If this fails, perform a local socket test on the Application Server to verify proper configuration.
A bridge test program is provided, dxbridgetest.dl4, on the Application Server.
Type “run dxbridgetest.dl4”, from the Application server’s home directory, and it will display the dynamicXport revision number if the Application server is configured correctly.

To do the most basic form of test, telnet to the socket on the Application Server, from the Application Server:
telnet 127.0.0.1 9632
This command telnets to the local host (127.0.0.1) on port 9632. If the inetd process is configured properly, the process should timeout and return a dynamicXport error message.

Initial Login
By default, after installation the login page can be displayed in the browser using the following address :
Domain name/dx/login/dxstd/login.html
Use the User ID *admin* and the password you defined when installing the Application Server.

If an error occurs, RECORD NOT FOUND, verify your inetd configuration is correct.

If you can login,the menu appears, but when selecting any menu option page presents 'HTML Page Not Found' error, verify that S_URL and S_SecureURL in dxserver.txt on the web server are set to the correct URL address. (be sure it begins with http:// or https://)

DXSYSTEM Parameters File

The DXSYSTEM file contains the global configuration parameter settings for DynamicXport. The file is a text file that is maintainable with any text editor. The file is located on the application server in the same directory as the listener program (/dx). The filename is dxsystem.txt.
The format of the file is Parameter Name = Parameter Setting.
The parameters are as follows :

REVISION = Do not modify!

DXBufSize = number of characters (16-bit Unicode character set) to send or receive through the socket in any one transmission. Default is 128000 characters. Minimum is 1024 characters. Decreasing can enhance performance, increasing can reduce performance. Should match DXBUFSIZE in dxserver.txt file.

Filetype = F=Foxpro Full-ISAM (default)
S=Microsoft SQL

UnixVerify = Reserved for future use

Allowed = Y/N Y=Allow access (default)
N=no access allowed until reset

Loginfields = Is an optional parameter. By default users login using their User ID (up to 8 characters) and Password. Defining this parameter makes it possible to login using any of the fields in the user file. The syntax for LOGINFIELDS is :
LOGINFIELDS= fieldname in user file AND/OR another field
The & symbol is the AND operator and the | symbol is the OR operator. The AND operator has a higher precedence than the OR operator. The expression is evaluated from left to right, but the precedence order may be modified by using parentheses.
The first defined field in LOGINFIELDS is used to select the index to search in the user file and does not support the OR operator..
The total length of LOGINFIELDS is limited to 254 characters.
The fieldnames begin with L_USER followed by the actual fieldname in the files/dxuser.dbf file (with the exception of password, which is simply password and User ID which is simply ID).
The alternative login method also requires customizing the login and verification html pages to submit the same fieldnames listed in the LOGINFIELDS parameter.
Examples:
LOGINFIELDS=ID & L_useremail
LOGINFIELDS=L_username & (password | L_useremail)
LOGINFIELDS=L_UserUserdef1 & password

Emailaddress = Is an optional parameter. If set, whenever a dyanmicXport error is logged in the Audit file, an email notification is sent to the email address specified here. The content of the email message is identical to the log file. The “AuditFileType” must also be set to either a “T” or an “F” in this dxsystem.txt file.

SMTPServer = Is an optional parameter to specify the SMTP email server name to use to send emails.

Readtimeout = # of seconds for Socket read timeout and application timeout (default 180)

Timeout	=	# of minutes of inactivity before the user must revalidate themselves with password (default 15) (range 0 to 24 hours or 1440 minutes)
Elapsed	=	# of hours session is continually active before the user must revalidate themselves with password (default 24) (range 0 to 41 days or 984 hours)
Expired	=	# of hours of inactivity before a session expires and is automatically logged off. (default 8) (range 0 to 41 days or 984 hours)
WebServer1IP	=	IP address of first web server
WebServer2IP	=	IP address of an additional second web server (could be used for an intranet or development web server)
WebServer3IP	=	IP address of third web server if needed
IPVerify	=	Y/N indicator, if yes, verifies that the Outside IP (Web Server's) address matches (default Y)
Bridge1IP	=	IP address of the first server permitted to access DynamicXport options through the DXBRIDGE program. The bridge porgram allows internal applications to perform DynamicXport options and receive results in a text file for further processing and reduces option security levels from high to medium when accessed in this manner. Do not configure unless used. A BridgeIP must also be listed as a WebServerIP.
PwdMinLength	=	The minimum required length of user passwords. The default value is 5.
PwdMaxLength	=	The maximum allowed length of user passwords. The default value is 20 and the maximum size password is 32.
PwdChances	=	# of chances to enter a valid password; if number is exceeded password is made invalid for period of time (default 5) (range 0-9)
PwdDelay	=	# of minutes to temporarily lockdown a user after too many failed login attempts (default 10) (range 0 to 24 hours or 1440 minutes)
UBProcessor	=	Processor to run Unibasic applications (default /usr/bin/unibasic)
dl4Processor	=	Processor to run dl4 applications (default /usr/bin/run)
GlobalExtendedLog=		0=off 1=on (default) IP#
		If on, does detailed Audit Log for all transactions. Log retains all input and output data. There is overhead related to Extended Log and it is recommended that it not be done Globally on a 'live' server. Optionally a specific IP address can be logged by setting to an IP#.
PurgeAuditAge	=	90 # of days to keep Audit records. Dxpurge.dl4 program must be run, typically as a cron job to purge old audit & session records.
CronFrequency	=	00:15 This parameter controls the frequency of purging dead sessions and audit records. The default value of 15 minutes is used if the CRONFREQUENCY

			setting is undefined or if it contains an invalid entry. The format is hours:minutes.
DefaultDir	=	dxstd/	Default Option directory to merge to if View cannot be determined
FilesDir	=	files/	Directory to find DynamicXport files on Application Server
Org	=	“dxorg”	Organizations (groups of users, ie departments, customers)
View	=	“dxview”	User visual interface options, assigned to Orgs
User	=	“dxuser”	Users within Organizations
Options	=	“dxoption”	Menu Options
GroupUsers	=	“dxgroupusers”	Reserved for future use
GroupOptions	=	“dxgroupoptions”	Reserved for future use
UserOptions	=	“dxuseroptions”	User’s menu options and news sections and mail/notify users
Contacts	=	“dxcontacts”	External contacts address books
Notify	=	“dxnotify”	Reserved for future use
News	=	“dxnews”	Reserved for future use
UserQueue	=	“dxuserqueue”	Reserved for future use
Audit	=	“dxaudit”	Usage audit log
AuditFileType	=	F=Foxpro Full-ISAM (default) Access provided by DynamicXport, supports extended logs T=Textfile (DynamicXport browser based view audit page will not access), extended logs not available. Blank = no audit file, auditing turned off	
Logins	=	“dxlogins”	Reserved for future use
Codes	=	“dxcodes”	Reserved for future use
Session	=	“dxsession”	Session State file

Testing of custom web pages and Unibasic/dL4 applications. Refer to Web Developers Brief and Application Developers Brief for explanation of testing process to test custom applications. After a custom application is tested stand-alone it can be published and tested through the DynamicXport communication layer.

Administration

Overview

The purpose of this section is to provide DynamicXport Administrator documentation.

Web directory structures

Refer to the Web Developer's Brief for information on the web site directory structure.

Relationships

Organizations/Users/Options Hierarchy

Organizations are a group of users such as customer, a supplier or a department.

(Future) Organizations can have additional Organizations belonging to it, creating a Parent, Child, and Sibling hierarchy of organizations. Unless overridden, Organizations receive the defaults and configuration of it's parent.

There is always one top level Organization that contains the system-wide defaults and configuration.

A user belongs to an organization and receives the defaults and configuration of that organization.

Options are various application programs and links that can be run from the user's menu.

Each user is assigned a list of options and can also be designated to additionally receive a copy of another user's options.

Organizations

Organizations define a group of related users. For example, you may define an Organization for all your internal staff members or you may define an Organization for each department. You may define an Organization for all your customers and you further define an Organization for each customer.
When each User is defined they must belong to an already defined Organization. An Organization can have any number of users belonging to it. Any number of those users can be assigned administrative privileges to administer the Organization and it's users.

The highest level of Organization is the '(MASTER)' Organization. The system administration user belongs to the '(MASTER)' Organization and has administrative privileges for the '(MASTER)' Organization and all the Organizations that belong to it. The '(MASTER)' Organization cannot be deleted.

All Organizations belong to the Organization hierarchy. The hierarchy starts with the '(MASTER)' Organization, then 1st level Organizations whose 'Parent' Organization is the '(MASTER)', then 2nd level Organizations whose 'Parent' Organizations are the 1st level Organizations. Thus, Organizations take on a parent-child relationship. The hierarchical structure provides a great deal of order and flexibility to the User privileges and administration model.

Organization file field definitions :

Organization ID
The Organization ID can be up to 8 characters in length and can consist of alphanumeric characters, comma, period, dash and underscore only. The ID is the unique identifier of the Organization and is case sensitive.

Parent Organization ID
The Organization ID to which the Organization is a 'child' of. First level organizations would have '(MASTER)' as their Parent Organization ID.

Organization Name
Up to 30 characters to describe the Organization such as a department name or company name.

Application ID
Up to 20 characters to define the existing applications code or ID for this Organization, such as a customer code, vendor code or company number. This can then be used by the custom application to tie an Organization to a particular application code. Where possible it is suggested to also use the existing application ID for your Organization ID.

View ID
Must be a valid View ID in the View Definitions. The View Definitions allow you to define different user interfaces. The interface that a particular user receives is then determined by which View ID is assigned to the Organization they belong to. The default View ID is 'dxstd'.

Prelink String
Application related, the Prelink field is a string that is to be processed before processing a dL4 or Unibasic Option Link. Typically this is used to set environment variables such as LUMAP and LUST prior to processing the option. This is a default Prelink for any option that is run by this Organization's Users. A Prelink can also be defined by Option in the Option definition. The Option Prelink string overrides the Organization Prelink string. The Prelink string can be up to 254 characters in length.

Exit URL
The Exit URL defines the URL (web page) to go to upon existing or logging off DynamicXport. It is typically a URL to a home page. If blank DynamicXport defaults to the (MASTER) Organization's Exit URL. The field is up to 60 characters in length.

IP Range
For enhanced security, users belonging to an organization can be limited to using DynamicXport only from defined IP's or IP ranges. If not within one of these IP 's or ranges the user will be denied access to the system. Full IP addresses must be entered. IP ranges are entered as beginning IP address, dash, ending IP address. Multiple definitions are separated by semi-colons.
An example is : 201.201.170.115-201.201.170.120;201.201.180.210
Up to 254 characters can be used to define the ranges.

IP Verify
This is an indicator which determines how a user is to be handled if their IP number changes during their DynamicXport session. This could be an indication of spoofing or it could be valid if the user is accessing via wireless or has a dynamic IP. Typically this would be defined as 'Deny' which would deny access if the IP number changes. ‘Validate’ would require the user to revalidate themselves and then continue the session.
A third option, ‘None’ disables the IP verification feature for this organization. This may be necessary for users accessing using dynamic proxy servers, causing incoming IPs to vary frequently.

Multiple Sessions
This is an indicator which determines if the user is permitted to have multiple simultaneous sessions.

Multiple sessions may be desired in the case of a shared or general login. If multiple sessions is not allowed and the Verify indicator is verify then a user cannot have multiple sessions but can resume an existing session from another location if the session is not expired.

User Defined 1-5
There are five 60 character fields available for storing information about the organization that would facilitate further linking the organization to the web page or application or to store information about the organization that is not available in the application.

Note: You cannot delete 'admin' user or change it's Organization ID or Administration level.

Views

View Definition records contain the parameters, directory paths and filenames which together define the view (or look and feel) that a user will experience through the browser. Once a View is defined it can be referenced by Organizations in the Organization definitions. Hence all users within that Organization would receive that View definition. Any number of views can be defined and any number of Organizations can reference a view.

The default predefined view ID is 'dxstd'.

View file field definitions :
View ID
The View ID can be up to 8 characters in length and can consist of alphanumeric characters, comma, period, dash and underscore only. The View ID must be a unique identifier for the view and is case sensitive.

Option Directory
The Option Directory defines where to find the HTML, images and icons to be used for menu option presentation and some other DynamicXport standard pages such as logoff.html and msg.html.
The default Option Directory is dx/dxstd/, which is defined in the view file as relative to the dx directory, or dxstd/
To create a new Options look and feel you can create a new Options theme directory by copying the dx/dxstd directory to another directory under dx/ and then modify the options files.
For example copy dx/dxstd/* to dx/siteabc/*. Then reference the new Option Directory in the View definition as siteabc/
Images and icons directories are subdirectories within the option directory.

Option Type
The Option Type indicated which type of menu option presentation to use. There are several pre-configured types that can be used and modified or you can create your own and reference it as Option Type x. Option Type 1 will utilize option1.html, type 2 option2.html, etc and Option Type x will utilize optionx.html

Option Type 1 - Top Frame/Drop down box
Uses a Top Frame with drop down list of options. Top frame has smalllogo.gif with an Alt/Link to Exit URL. Background can be changed in .css class= OptionTopFrame and class=OptionTopFrameSub and class=OptionBody.
(By default, main page has the largelogo.gif centered.)
(It is recommended that Option Descriptions be kept below 18-20 characters for proper display.)

Option Type 2 - Left Frame/Vertical list
Uses a left frame with a list of options. Left frame has smalllogo.gif with an Alt/Link to Exit URL. Background and fonts can be changed in .css class= OptionLeftFrame and class=OptionLeftFrameSub and class=OptionBody

(By default, main page has the largelogo.gif centered.)

Option Type 3 - No frame/Full page
Uses the full screen to display options (typically as large icons). Uses top3.gif for a heading with an Alt/Link to Exit URL and left3.gif for side image. Background of page can be changed in .css class=OptionBody

---

Option Type 4 - Top frame/Horizontal tabs
Uses a Top Frame with foldertabs of options. Background can be changed in .css class=OptionTopFrame and class=OptionBody

---

Option Type x - Custom
Custom defined/designed HTML options page

Menu show type
This indicator defines how each menu option is presented to the user.
Choices are :
1. Display Option Description
2. Display Option ICON image

If displaying ICONs for options, an icon image file must be created for each menu option and placed in the Option Directory icons subdirectory.

Merge Directory
The merge directory defines where to find the html pages to merge to for all the custom applications.
The Merge directory field is available to the application as an S_ field (S_VIEW.MergeDIR) to facilitate the application merging to the correct html page. The Merge directory automatically precedes an Option's Input page and Output page.
The Merge Directory allows for the flexiblility of defining different View definitions and HTML pages to facilitate different stores, locations and language variations all using the same DynamicXport platform and same applications.
The Merge Directory should be defined relative to the dx/ directory and should be at the same directory level as the dx/cgi-bin directory. For example, the directory could be /dx/custom/ and referenced in the View definition as custom/

Stylesheet filename
The stylesheet filename defines the stylesheet .css filename to use for this view. The stylesheet file is a centralized file on the web server which defines various web presentation parameters such as fonts, colors and backgrounds. The default stylesheet file is dx/styles/dxstd.css and is referenced as styles/dxstd.css
To create another scheme or template copy the dxstd.css to another filename, modify the definition and then reference it in the view definition.

Messages filename
The message filename defines the application messages text filename to use for this view.
The message file is a centralized file on the application server which defines the message text to display for associated message numbers. These are typically used but not limited to error messages.
The format of the file is message #, colon, message text. Message text is limited to 74 characters. Message numbers 0-10000 are reserved for DynamicXport. Message numbers above 10000 are available for your application's use.
The message file MUST reside in the same directory as the dx listener program. The default message file is dx/dxmsg.txt and is referenced as simply dxmsg.txt in the View definition.
To create another message file for different languages, etc. copy the dxmsg.txt to another filename in the same directory, modify the messages in the file and then reference it in the View definition.

Users

Each user login to the DynamicXport platform is defined in the User definitions file. Each user must belong to an already defined Organization. When a user logs into the system, they are verified against the Users file and then presented with their options defined in the User Options file based on the View ID referenced by their Organization.

DynamicXport installs with one predefined administration user. The administration user can then create other users and give them administrative privileges if desired.

User file field definitions:
User ID
The User ID can be up to 8 characters in length and can consist of alphanumeric characters, comma, period, dash and underscore only. The ID is the unique identifier of the User and is case sensitive. Typically a part of a person's name, initials, customer code or other unique recognizable identifier is used.

Organization ID
Identifies the Organization that this User belongs to and thus determines the User's level in the Organizational hierarchy.

Password
This is the User's secretive password to access the system. The password must consist of alphanumeric characters only and must be at least 5 characters in length. The password can be up to 20 characters.

User Name
Up to 30 characters to describe the User, typically first and last name.

Application ID
Up to 20 characters to define the existing applications code or ID for this User, such as a customer code, salesman code, employee number, etc. This can be then be used by the custom application to tie a User to a particular application code.

Administration Level
This indicator defines what DynamicXport administration privileges this user can access.
If none, the user has no administrative privileges.
If Organization, the user has administrative privileges within their Organization and it's descendants (Organizations under their Organization). They can add, change and delete users and assign options to users.
If Global, the user has administrative privileges within all Organizations.

Extended Log
This indicator defines whether an Extended Log should be saved for this user. If this indicator is turned on the Audit Log will retain extensive details of all incoming and outgoing data. This can be used by developers for debugging purposes or for extensive monitoring of a particular User. Extended Log can also be turned on Globally or by selected Option.

E-mail address
This field is used to store the user's e-mail address if known.

User Defined 1-5
There are five 60 character fields available for storing information about the organization that would facilitate further linking the organization to the web page or application or to store information about the organization that is not available in the application.

Limited Users

This option is defined as a main menu option and it's html is located in the dxstd directory.
It has the same functionality as the Users option except it is limited in scope as to what fields in the User file can be viewed and maintained. The purpose of Limited Users is to provide a method for Organization Administrators to maintain their Users without full understanding or knowledge of DynamicXport. It also will use the look of the VIEW.OptionDIR so the look will match the custom application.

Automatically Generated User IDs

If the User option or Limited Users option html is modified to allow a null User ID to be submitted on a User Add action, the User ID will automatically be generated. The ID will be the next highest number between 00000001 and 99999999.

Options

Each menu option to be accessed by a user is defined in the Options definition file.
Based on the Link type the option can be a link to a Menu title, a URL, a dL4 application or a Unibasic application.
Once an option is defined it can be added to Users' option lists so it can be accessed in the User Maintenance page.

DynamicXport installs with all it's administrative functions predefined as Web Administration options.

Option file field definitions :
Option ID
The Option ID can be up to 8 characters in length and can consist of alphanumeric characters, comma, period, dash and underscore only. The ID is the unique identifier of the Option and is case sensitive.

Organization ID
Identifies the Organization that this Option belongs to and thus determines at what level in the Organizational hierarchy the Option is available. Typically Options are placed in the (MASTER) Organization and thus available to be assigned to all users. However Options can be placed at a lower level and therefore only made available to User's in that Organization or it's child Organizations.

Parent ID
Options can be presented on the User's menu option list as a sub-option below a menu title. First define an Option to be a menu title and then reference it as the Parent ID when defining the sub-option. The Parent ID must already be defined as a Menu Link type option.

Description
This field contains the description of the option that will appear in the User's menu options list. The description can be up to 40 characters.

Link type
This indicator defines the type of link the option is. Valid link types are :

1. Menu - is simply a menu title to display on the User's menu. Other options can then be defined as sub-options of the menu option by using the Menu Link type Option ID as it's Parent ID.
2. URL - simply links to the URL defined in the Link field when the option is selected by the user.
3. dL4 - indicates it is a dL4 application that is to be run when the option is selected. The dL4 program to run is defined in the Link field.
4. Unibasic - indicates it is a Unibasic application that is to be run when the option is selected. The Unibasic program to run is defined in the Link field.

HTML Input Page

If an initial HTML page needs to be displayed before running the application the filename of the HTML page is put here. This is typically a form input page which would be displayed when the Option is chosen on the menu. When the form is submitted then the application is actually run to respond to the form inputs. Session variables can be merged into the page.
This field will ALWAYS be prefixed with the View Merge Directory.
For URL link types, if the Input Page field is used a merge output will be done to the page (the page must be in the Merge Directory) vs. if the Link field is used a redirect to the page will occur. The redirect will assume the page is in the Merge Directory unless a full URL is given.

Prelink String
Application related, the Prelink field is a string that is to be processed before launching the dL4 or Unibasic Option Link. Typically this is used to set environment variables such as LUMAP and LUST prior to processing the option. The Prelink string can be up to 254 characters in length. If not defined, Options will use the Prelink String defined for the Organization.

Link
The Link field is the URL to link to or the dL4 or Unibasic program filename to run.

HTML Output Page
The HTML Output Page is the default page to merge results to and display to the browser after the application has finished processing. The Output Page can be overridden by setting the Output= variable to another page. DynamicXport automatically precedes the HTML Output Page field with the View Merge Directory.

Target Window
The Target Window defines where the Option is to display in the browser.
If set to 'Same', it will display within the same window browser as the menu options.
If set to 'New', it will always open a new separate window.
Any other text will be the name of a window, in which case if it exists it will display in it, otherwise it will create a new window of that name.

Requires SSL access
This indicator determines if the option should be run with SSL encryption. Menus will hyperlink to the option with http or https based on this indicator. If SSL is required DynamicXport will verify that it is being accessed with SSL before allowing access.
NOTE: SSL is not required when accessing the option through the bridge driver.

Access Security Level
This indicator determines in which ways the option can be accessed.
If High, the Option must be accessed by a User who has already logged in and has an active session and the option is on the User’s options list..
If Medium, the Option can be accessed with an active session or directly with a User ID and password. The option must be on the User’s options list.
This could apply where someone else's application is trying to access data through a socket instead of through a browser.
If Low, the Option can be accessed and run with or without an active session or a User ID and password.

Extended Log
This indicator defines whether an Extended Log should be saved for this option. If this indicator is turned on the Audit Log will retain extensive details of all incoming and outgoing data. This can be used by developers for debugging purposes or for extensive monitoring of a particular Option. Extended Log can also be turned on Globally or by selected Users.

User Options

The User Options page allows an Administrator to assign menu options to a user. When the user logs in their assigned menu options will be presented for access. An administrator can also arrange the order in which the user's options are presented by moving an option up or down on the user's assigned list.

The User Options page can be accessed by clicking on 'Options' under More Info column in the Users list. Also, when a new user is added the Add Options button can be clicked on.

The page will display two menu list boxes. The list on the left contains all the menu options that the Administrator can assign to the user that the user is not already assigned. The list on the right contains the menu options that are currently assigned to the user.

To assign an option to the user, click on the option to add in the left box and then click on the > box to move it to the right box.

To remove an option from the user, click on the option to remove in the right box and then click on the < box to move it to the left box.

To assign all the available options to the user, click on the >> box to move all the options in the left box to the right box.

To remove all options from the user, click on the << box to move all the options in the right box to the left box.

If you select a Menu Option Title, ALL the options listed under the title will be moved. To add or remove selected items under an Option Title select those items individually.

To rearrange the User's Option list, click on the item you want to move and then click the up or down box. Clicking on a Menu Option Title will move all the options under that title. To move an item within an Option Title select the item individually. All items under a Menu Option Title must remain grouped together, and thus cannot be split.

When finished configuring the user's option list, click on Save to update their option access list.

In addition to assigning menu options to a user, an Administrator can assign a user a duplicate copy of another user's option list. This allows an Administrator to quickly assign a standard list of options to a user. (If the list of the copied User is updated it will be modified for the User it is copied to as well) The User ID that the Administrator selects to copy must be in the same Organization or a child Organization of the user it is being copied to.

View Sessions

View Sessions allows you to view currently active User Sessions. The purpose of the page is to allow an Administrator to see who is currently active on the system, how long they have been logged on, and their current session variable values. By default the page will display a list of all the active sessions. You can enter a selected User ID to view only the sessions for that user.
Clicking the ‘Show Session Count’ checkbox on will display a total count of active sessions (sessions not yet expired).
Clicking on the ‘Delete’ icon will terminate that particular session.
The columns that are shown for each session are :

Session ID	The ID that is automatically assigned to each user session by DynamicXport.
IP number	The IP number of the User that is logged into this session.
Started	The date and time this session started. The time is based on the time of the application server.

Last	The date and time of the last application server access by this session.
Org ID	ID of Organization user belongs to.
Name	Organization name
AppID	Organization App ID
User ID	ID of the User logged in.
Name	User name
AppID	User App ID
More Info	Clicking on S_Fields will display a table of the User's S_Fields, which are standard DynamicXport session variables. Clicking on U_Fields will display a table of the User's U_Fields, which are User-defined DynamicXport session variables.

To view what a user has accessed and/or detailed transactions, past or current, use View Audit.

View Audit

View Audit allows you to view past and current application server accesses. The page can be used during development or debugging to view the inputs, outputs and errors that occurred during processing.
The page can also be used by an Administrator to view who accessed what options and when. This can be helpful in determining how much a user accesses the system and what options they access.

You can narrow the Audit list that is viewed by selecting a particular User ID, Option ID or IP. You can select an additional User ID, Option ID or IP with a logical AND or OR condition to further narrow the list. For example you can specify to view only accesses by User ID 'Bob' and Option ID 'Orders'.
A 'From Date' and 'To Date' can be specified to limit the list to a desired time period.

If the 'Only if S_Msgno' checkbox is checked only those accesses that recorded an error will be counted or shown. This is helpful for ongoing monitoring and debugging.

If the 'Only Action=Input' checkbox is checked only those accesses where the Action field is equal to 'Input' will be counted or shown. These would typically be when the Option is chosen from the User's Menu Options and provides a more accurate count of User and Option usage.

When viewing the audit file, if interested in statistics only, you can use one of the Summary List options.
Summary by Option will provide a Count of the number of times each Option ID was accessed.
Summary by User will provide a Count of the number of times each User ID accessed.
Summary by User & Option will list each User and each Option they accessed and a Count of the number of times they accessed each of them.

The detailed audit list can be displayed in ascending or descending sequence.

The columns that are shown for each access are the User ID, Organization ID, IP, Option ID, Start (date/time), Action, S_Intmsg, S_Msgno and Extended Log. Scrolling over the User ID or Organization ID will display the User name and Organization Name, respectively. Scrolling over the S_Msgno column fields will display the error message for the message number. If an Extended Log was processed for a transaction, the Extended Log column can be clicked to view the Extended Log Input and/or Output fields and values. Extended Logs can be turned on globally for all users or a selected IP #, as well as by User or Option.

Troubleshooting

Socket

To verify that the socket from the web server to the application server is functioning, from the browser use the following address :
Domain name/dx/cgi-bin/dxserver.cgi?Option=Revision
The browser will display the revision number of DynamicXport if the socket is configured properly.

Audit File

When developing or troubleshooting reported problems the DynamicXport Audit file is the first resource to verify a process is occurring as expected and no errors are reported by the listener or the application.

Browsers

Setting You Browser to Allow Javascript

DynamicXport requires that Javascript is enabled on the client's browser. To enable JavaScript in Microsoft Internet Explorer® and Netscape® browsers, see your operating system's user manual or on-screen Help directory—or follow the instructions below.

Internet Explorer 4.0:

1. Click on "View" in your browser's menubar, then choose "Internet Options" from the drop-down menu.
2. On the "Internet Options" screen, click on the "Security" tab.
3. Select "Internet Zone" in the "Zone" box, then select "Custom" and click on the "Settings" button.
4. Scroll down to "Scripting," then to "Active Scripting."
5. Make sure the "Enable" check box is selected.
6. Click on the "OK" button.

Internet Explorer 5.0:

1. Click on "View" in your browser's menubar, then choose "Internet Options" from the drop-down menu.
2. On the "Internet Options" screen, click on the "Security" tab.
3. Select "Internet Zone" in the "Zone" box, then select "Custom" and click on the "Settings" button.
4. Scroll down to "Scripting," then to "Active Scripting."
5. Make sure the "Enable" check box is selected.
6. Click on the "OK" button.

Netscape Browsers:

1. Click on "Edit" in your browser's menu bar; then choose "Preferences" from the drop-down menu.
2. On the "Preferences" screen, click on "Advanced" in the left column.
3. Make sure the "Enable JavaScript" check box is selected.
4. Click on the "OK" button.

If select users receive a 'Page Expired' message when using the Back button, likely their browser cache is full and needs to be cleared.

If select users are repeatedly asked to revalidate or receiving Session Activation Error pages, their browser IP may be dynamically changing. You may need to reduce the security level of the organization they belong to by changing the IP Verify field to 'None'.

DynamicXport Error Messages

Listed below are more common DynamicXport error messages and possible causes.

If when a particular option is chosen and then the browser appears to hang and then times out, this indicates the Option's program is accessible and attempted to run and then incurred an internal error. This could be caused by attempting to open a file that cannot be found, a file that is not read or write accessible by all users, or chaining or calling another program that cannot be found or is not read accessible or is not saved under the required Unibasic or dL4 version.

NO DATA FROM APPLICATION SERVER
If this occurs at login or with all options most likely the Application Server is not responding to socket requests. Try the Option=Revision test to see if a response is received. Verify the Application Server is accessible.
If this occurs immediately when a particular option is chosen the program defined for the option could not run or timed out, verify the Option Link (program) is named properly, the Prelink (LUST,LUMAP) is defined properly and the program is read accessible by all users.
This error can occur if the application system is running out of message queues. A minimum of two message queues is needed for each simultaneous web access. Refer to Passport FAQ for more information on message queues.
You can also view the Audit file for more details.

8002: PRODUCT REVISION ERROR
This error indicates that the DynamicXport revision on the Web Server does not match the DynamicXport revision on the Application Server. The two server revisions must match to function properly.

8007: LOGIN FAILURE DUE TO INVALID USER ID OR PASSWORD
This error occurs when logging into DynamicXport and the User ID or password entered are incorrect.

8013: NO SUCH OPTION
This error occurs if attempting to run an Option which is not defined in DynamicXport. Option ID's are case sensitive which may be the cause of the problem.

8014: SESSION IS NOT OWNED BY THIS ACCOUNT
This error occurs when revalidating a session with an account User ID other than the one originally logged in as.

8015: INVALID PASSWORD
This error occurs when revalidating and the password entered is incorrect for the User ID.

8018: INTRUDER DETECTED

8019: OPTION REQUIRES SSL CONNECTION
This can be caused if attempting to access an Option which is configured with SSL Required, such as Change Password, using standard HTTP.

300: IP VALIDATION ERROR
This can occur if the user's IP number has changed during access. If the user is accessing using Satellite Access or other types of wireless access where the IP may change, the IP Verify indicator for their organization may need to be changed from 'Deny' to 'Revalidate'.

400:APPLICATION DIED UNEXPECTEDLY
The application program could not run. Check paths, filename and protection.

400:APPLICATION DID NOT COMPLETE
The application program is not completing or did not perform a DXCLOSE statement.

100 Record not written
Possible problem reading socket, possibly application server is not available or accessible.

Illegal filename File not found
Possibly S_Root is not set properly in dxserver.txt on the web server.

FILE NOT FOUND
Indicates it could find the html file to merge to on the web server.
To display the filename that could not be found, set SHOWOUTPUTFILE=Y in the dxserver.txt file on the web server.

OUTPUT PAGE NOT SPECIFIED
Indicates that the output page to merge to was not defined in the Option definition and was not defined by setting the 'Output' field.

Unexpected Error in DynamicXport
Unexpected, thus there is no indication of what may be causing the error. Verify the application functions in standalone test mode and that the web server and application server are communicating properly.

Appendix A Directory Lists

HTML Files

Standard HTML files delivered with DynamicXport

Filename	Purpose
dx/admin/dxorgpr.html	Organization List
dx/admin/dxorgmnt.html	Organization Record Maintenance
dx/admin/dxorghelp.html	Organization Help
dx/admin/dxuserpr.html	User List
dx/admin/dxusermnt.html	User Record Maintenance
dx/admin/dxuserhelp.html	User Help
dx/admin/dxuseropmnt.html	User's Options List & Maintenance
dx/admin/dxuserophelp.html	User's Options Help
dx/admin/dxoptionpr.html	Options List
dx/admin/dxoptionmnt.html	Options Record Maintenance
dx/admin/dxoptionhelp.html	Options Help
dx/admin/dxoptionusers.html	Users who have access to selected Option window
dx/admin/dxviewpr.html	View List
dx/admin/dxviewmnt.html	View Record Maintenance
dx/admin/dxviewhelp.html	View Help
dx/admin/dxsessions.html	Display current active sessions and status
dx/admin/dxsessionfields.html	Display session and user-defined session fields and values
dx/admin/dxsessionshelp.html	Display sessions Help
dx/admin/dxauditpr.html	Display audit file
dx/admin/dxauditfields.html	Display extended log fields and values
dx/admin/dxauditprhelp.html	Display audit Help
dx/admin/dxmsg.html	Displays DynamicXport Administration message pages
dx/login/dxstd/login.html	Login screen
dx/dxstd/option.html	Option forwarding after login
dx/dxstd/option1.html dx/dxstd/option1help.html	Options display, type 1, top frame
dx/dxstd/option2.html dx/dxstd/option2help.html	Options display, type 2, left frame
dx/dxstd/option3.html	Options display, type 3, large icons
dx/dxstd/msg.html	Displays DynamicXport messages

dx/dxstd/logoff.html	Displays DynamicXport Logoff page
dx/dxstd/passwd.html	Displays Change Password page
dx/dxstd/validate.html	Displays user revalidate page
dx/dxstd/dxtest.html	Displays Begin HTML test page for development
dx/dxstd/blank.html	Displays blank page within frame (initializes Dynamic frames)
dx/dxstd/redirect.html	Redirects to Options URL
dx/dxstd/dxuserprl.html	User list with limited search/list fields and OptionDIR look
dx/dxstd/dxusermntl.html	User Record Maintenance with limited fields and OptionDIR look
dx/dxstd/dxuserhelpl.html	User help, limited to fields in dxusermntl

dx/examples/cust.html	Example page, Simple Customer name display
dx/examples/custpr.html	Option 'EXCUST' example first page, Multipage Customer
dx/examples/custmnt.html	Option 'EXCUST' example second page
dx/examples/doconfirm.html	DODEMO, confirm order received page
dx/examples/dohelp.html	DODEMO, help text page
dx/examples/doordermnt.html	DODEMO, maintain item on new order page
dx/examples/doorderpr.html	DODEMO, new order review and entry page
dx/examples/dostatpr.html	DODEMO, existing order search/display page
dx/examples/doprodmnt.html	DODEMO, maintain product page
dx/examples/doprodpr.html	DODEMO, product search/display page
dx/examples/forgotpd.html	Forgot Password page
dx/examples/newuser.html	Request a Login page
dx/examples/images	Directory of Images for example pages
dx/examples/prodimg	Directory of Product images for DODEMO
dx/examples/prodspec	Directory of Product Specs PDFs for DODEMO
dx/examples/dxcalexample.html	Example page, utilizing the dxcalendar javascript popup calendar

Includes Directory List

The Includes directory contains common HTML and Javascript that is included in HTML pages.
They can be used by the web developer within their custom applications but should not be modified.

dx/includes/

Filename	Function
dxcalendar.html	HTML used in conjunction with dxcalendar.js to provide a popup calendar.
dxcalendar.js	Javascript routines used in conjunction with dxcalendar.html and examples/dxcalexample.html to provide a popup calendar.
dxcopyright.html	Displays copyright notice, typically inserted at the bottom of pages
dxdate.js	Contains date verification, reformatting, displaying and compare routines.
dxemail.js	Verifies field is in a valid e-mail format.
dxhelp.js	Opens a standard help window and presents a help page in a named or unnamed window
dxmenu.js	Presents menu options, used by option1.html and option2.html
dxpopup.js	Used to display a URL in a existing or new target window. Function DX_popup(URL,target,features) Displays URL in the target window. If URL is an HTML string (starts with <), displays the HTML string in the target Window. (Javascript addcgi is necessary in the html including dxpopup.js. If URL begins with 'session=', URL is preceeded with dl4v(S_CGI) and appended with Action=Input to simulate an option selected from a menu.) If target is not sent defaults to current window If window features is not sent defaults to browser defaults Function DX_confirm(prompt) Displays a confirmation box containing the prompt. Function DX_printpage() Prints the current page if the browser supports it.
dxutil1.js	Contains commonly used functions DX_trim to remove left and right white space from a field and DX_required to trim and verify a field is not blank.
dxutil2.js	Contains commonly used editing funcitons. Provides functions to edit check phone numbers, zip codes, alphanumeric, integer, non-negative integer, integer within range, floating point numbers, floating points within range, remove specified characters and reformat string.
dxutil3.js	Contains function DX_setselect to set the selected value of an option list and DX_getbutton to get the value of a selected radio button.
formchek.js	Various Javascript editing routines

Images Directory List

dx/dxstd/images and dx/admin/images

Images in the dx/admin/images directory are used by the DynamicXport Administration pages and should not be changed.
Images in the dx/dxstd/images directory are used by User Option pages and standard template pages.

When there is a group of gifs with the same name, but numbered it is variations of a gif, where the gif will be selected based on the VIEW Object's Type Code (which will be merged into the HTML to create the correct filename). With these files the ALT should all be the same.

Filename		**Purpose**
dxtopd.gif	Default top gif if View Option Type not known and logoff	Default top gif
dxtop1.gif	Blank gif because Option 1 is top frame	Option Type 1 top gif
dxtop2.gif	Narrow band top, Option 2 is left frame	Option Type 2 top gif
dxtop3.gif	Full top, Option 3 is full screen menu	Option Type 3 top gif
dxtopx.gif	Full top, default to copy of dxtop3.gif, customize to whatever	Option Type X top gif

Filename typically coded as dxtopdl4v(S_VIEW.OptionType).gif
Alt : Return to Options Menu Link : dl4v(S_CGI)?Session=dl4v(Session)&Option=Options
Alt : Exit Back to Home Link : http://dl4v(S_VIEW.Exit.URL)

Filename		**Purpose**
dxleftd.gif	Default left gif if View Option Type not known and logoff	Default left gif
dxleft1.gif	Narrow band , Option 1 is top frame	Option Type 1 left gif
dxleft2.gif	Blank gif because Option 2 is left frame	Option Type 2 left gif
dxleft3.gif	Full left , Option 3 is full screen menu	Option Type 3 left gif
dxleftx.gif	Full left, default to copy of dxleft3.gif, customize to whatever	Option Type X left gif

Filename typically coded as dxleftdl4v(S_VIEW.OptionType).gif

---

DynamicXport Navigation buttons

Filename	**ALT**	**Purpose**
dxsearch.gif	Search by xxx	Search file button
dxadd.gif	Add New Record	Add New record button
dxsave.gif	Save Record	Save record button
dxdelete.gif	Delete Record	Delete record button
dxcancel.gif	Cancel	Cancel entry button

dxback.gif	Back	Back button
dxcontinue.gif	Continue	Continue button
dxgo.gif	Go	Go button
dxhelp.gif	Help	Links to help for the option
dxlogoff.gif	Logoff	Logs user off button

Miscellaneous

Filename	**ALT**	**Purpose**
dxarrowdown.gif		
dxarrowup.gif		
dxcal.gif		Calendar icon
dximgdelete.gif		Delete icon
dxport.gif		Powered by logo
spacer.gif		Horizontal spacing
spacerv.gif		Vertical spacing

Etc.

DynamicXport Logos

Filename	**ALT**	**Purpose**
smalllogo.gif		Options page small logo
Alt : Exit Back to Home	Link : http://dl4v(S_VIEW.Exit.URL)	
largelogo.gif	DynamicXport	Options large logo

Option Selections ICONS :

/dx/dxstd/DXUSER.gif		User Maintenance
/dx/dxstd/DXUSERimg.gif		Organization Maintenance
		Options Maintenance
		View Maintenance

Etc.

---

dx/login/ directory

logintop.gif		Login top image
loginleft.gif		Login left image
login.gif	Enter User ID and Password, then Click here!	Login button

Appendix B Stylesheet Classes

User-defined Styles

See the /dx/styles/dxstd.css file for latest definitions and default values.

OptionTopFrame
Used to present Options Top Frame (in frame's <BODY>) (also need LINK in frame's <HEAD>)
OptionLeftFrame
Used to present Options Left Frame (in frame's <BODY>)
OptionBody
Used to present Options Main Screen Body
Title
Used to present any Page titles
TableHeader
Used to present any Table Column Headers
TableCell
Used to present any Table Cells
TableBorder
Used to present any Table Borders
FieldLabel
Used to present any Field Labels
FormBorder
User to present any Form Borders
InputField
Used to present any Input Fields
LoginLabel
Used to present login screen field labels only
Note
Used for notes
Footer
Used to present footer information, ie copyright
Error
Used to present any Errors

Standard HTML tags defined

Body
Used as body default
A:link
A:visited
A:active

Appendix C Session Variables

DynamicXport Session Variables are automatically stored for each active user session. Most of the fields do not change throughout the session. The fields are managed by DynamicXport and most cannot be modified by the application.
The fields can be merged into your HTML pages as a dl4v value. For example, the merge the user's name onto a page simply place dl4v(S_USER.Name) in the html.
The files can also be accessed by the application using the DXGET statement. For example, to find out the user's application ID use the statement DXGET("S_USER.AppID",App$).

Session=	Current session #
Option=	Current option chosen
S_Root=	The root directory on the web server of the dx/ directory. Is defined in dxserver.txt file on web server and sent through the socket.
S_CGI=	The CGI program to use on the web server. Is defined in dxserver.txt file on web server and sent through the socket.
S_URL=	URL to use for unsecure links. Is defined in dxserver.txt file on web server.
S_SecureURL=	URL to use for secure links. Is defined in dxserver.txt file on web server.
S_ClientIP=	The IP# of the user which is sent from the web server through the socket.
S_WEBIP=	The IP# of the web server connecting through the socket.
S_Revno=	The revision number of DynamicXport
S_ContentOut	Indicates the type of content to be sent out. Typically HTML. May be WML if user's web browser supports WML and Checkbrowser=WML is present in dxserver.txt file or S_ContentOut is explicitly set in dxserver.txt. This value may be used in an application to determine the output file to merge with.
S_ORG.ID=	Organization ID of user session
S_ORG.Name=	Organization name from Organization file
S_ORG.AppID=	Application ID from Organization file.
S_ORG.ExitURL=	The Exit URL defined for the User's Organization. If no Exit URL is defined for the Organization (null), the Exit URL defaults to the (MASTER) Organization's ExitURL.
S_ORG.UserDef1=	Organization file User defined 1
S_ORG.UserDef2=	Organization file User defined 2
S_ORG.UserDef3=	Organization file User defined 3
S_ORG.UserDef4=	Organization file User defined 4
S_ORG.UserDef5=	Organization file User defined 5
S_VIEW.ID=	View definition ID being used (referenced by the Organization)
S_VIEW.OptionDIR=	Option directory to find HTML pages to merge with as defined in the View definition
S_VIEW.OptionType=	Type of option page to display. This is a code which determines the option page to merge with in the Option directory.
S_VIEW.BmenuShow=	This is a code to indicate how to present the options within the option type.
S_VIEW.MergeDIR=	Merge directory to find HTML pages to merge with as defined in the View definition
S_VIEW.Stylesheet=	The stylesheet (.css) file to use when presenting pages as defined in the View definition
S_VIEW.Msgfile=	The message text file on the Application Server to use as defined in the

	View definition
S_USER.ID=	User ID
S_USER.Name=	User name
S_USER.AppID=	User App ID
S_USER.Email=	User email address
S_USER.Userdef1=	User file User defined 1
S_USER.Userdef2=	User file User defined 2
S_USER.Userdef3=	User file User defined 3
S_USER.Userdef4=	User file User defined 4
S_USER.Userdef5=	User file User defined 5

The following session fields can be set by your application and have special functionality within the listener :

S_Msgno=	You can set S_Msgno to a message number. If S_Msgno contains a value, DynamicXport will look up the message number in the message text file and overwrite S_Msg with the message text. The S_Msgno value is also always stored in the Audit Log, even if Extended Log is not on.
S_Msg=	You can set S_Msg to a text message to be merged to a page like any other variable. S_Msg is special in that if S_Msgno is set it will be overwritten with the S_Msgno's text.
S_Intmsg=	You can S_Intmsg to a text message that can be, but is not typically merged to the web page. S_Intmsg is special in that if S_Intmsg is set it will be stored in the Audit Log, even if Extended Log is not on. This field is intended to be used to track internal programming errors without displaying the error details on the web page.

Appendix D Examples

Typical Process Flow of an Option

Option is selected from Option Menu (often a frame with a target=”main”) (if the options window flag is set to new, the target will be a new window)

The Option on the menu is coded as an a href to dl4v(S_CGI)?Session= &Option= &Action=Input

DynamicXport passes form information through a socket to the application server.

DynamicXport :
Finds Option record in Options file
Opens session file & re-verifies IP.
Re-verifies option can be run.
Verifies session should not be timed out. If timed out outputs error page or validate page.
If Type=Input logs last accessed & # of times accessed in option file.
Based on Linktype, runs option as dl4, unibasic or system program as a bidirectional pipe.
The application provides output information with a Call DXSET.
When Application process is done DynamicXport saves U_ fields in the session file, passes results through the socket, and merges the results into the ‘Output’ HTML page.
It automatically merges hidden values for Session= and Option=

Example of a Simple Option

Display Customer Record
Purpose: To display a customer's record from a customer file. Assume that the User's AppID (Application ID) field in DynamicXport has been set to the customer ID code.

Unibasic code :

```
 90 Dim V$[20],N$[26]
100 Open #1,"Customers"
110 Call $DXOpen                        !Open DynamicXport to get field values
120 Call $DXGet,"S_USER.AppID",V$       !Read in value of S_USER.AppID into V$
200 SEARCH #1,2,1;V$,V1,V2              !Search for customer record
210 READ #1,V1;N$;                      !Read customer name
220 Call $DXSet,"Name",N$               !Set value for field 'Name' to merge to HTML
230 Call $DXClose                       !Proceed to merge fields to HTML page defined in
                                         DynamicXport Option record
```

HTML code :

```
<HTML>
<BODY>
The customer name is dl4v(Name)
</BODY>
</HTML>
```

Example of a Multipage Option

Customer Display/Search/Maintenance
Purpose : Will allow a search for customers by customer ID or Name, add a new customer, or select a customer to maintain/delete.
(This example is included in your installation)

Setup :
Definition of Option in Options file (dL4 version) :

OPTION.ID	EXCUST
OPTION.Desc	Customer Maintenance
OPTION.Linktype	dL4
OPTION.Input	examples/custpr.html
OPTION.Prelink	LUMAP="examples=/home/dxport/examples"
OPTION.Link	examples/excust.dl4
OPTION.OutputMerge	examples/custpr.html

Definition of Option in Options file (Unibasic version) (not configured at install, you can change EXCUST option or add a new option with this definition) :

OPTION.ID	EXCUST
OPTION.Desc	Customer Maintenance
OPTION.Linktype	Unibasic
OPTION.Input	examples/custpr.html
OPTION.Prelink	LUST=/usr/lib/ub:/home/dxport/examples ISAMSECT=8
OPTION.Link	excust.ub
OPTION.OutputMerge	examples/custpr.html

Option is selected from menu, and is passed to dxserver which displays the OPTION.Input page. Then the user can choose to add a new customer or search for a customer by customer ID or name. When they click a submit button the inputs are passed to dxserver which passes the inputs through a socket and runs the excust.dl4 application program. Inputs are read in the excust.dl4 application by a Call DXGet command.
excust.dl4 application logic :

If Action=Search (it is coming from the search form)
Keyvar will be the search value to exact or partial matches. Create new table of all Customers with a full or partial match to the Keyvar passed. Do this by creating string arrays of each field (column) that will display in the table. Show 40 records max at a time, set Keynext=next key to search if there are more than 40. Then do a Call DXSet command for each table column to merge the field arrays into the arcustpr.html page.

If Action=Add (it is coming from the search form as an Add New)
CUSTID will be the search string, sent as CUSTID. Verify if already on Customer file. If already on file set msg=error message and OUTPUT to std msg html file and end.
If not on file, set CUSTID to null and all customer fields to null or defaults, set CUSTNewID to the new ID, set OUTPUT to examples/custmnt.html

If Action=Change
CUSTID will be sent. Read customer record, set all the fields equal to the record, set CUSTNewID to the ID, set Output to examples/custmnt.html

If Action=Save
Save the new or changed customer record. If CUSTID is null it is an add record.
Set Msg=saved or error message and Output to msg.html

If Action=Delete
Delete the CUSTID record.
Set Msg=deleted or error message and Output to msg.html

HTML Code for custpr.html :

```
<html>
<body>
<table>
  <form method="POST" action="dl4v(S_CGI)">
        <tr>
          <td>Customer ID :</td>
          <td><input type="text" name="CUSTID" size="8" maxlength="8"></td>
          <td><input type="submit" name="Action" value="Add"></td>
        </tr>
  </form>
  <form method="POST" action="dl4v(S_CGI)">
        <tr>
          <td>Customer ID or Name :</td>
          <td><input type="text" name="Keyvar" size="30" maxlength="30"></td>
          <td><input type="submit" name="Action" value="Search"></td>
        </tr>
  </form>
</table>
<table>
   <tr>
     <th>ID</th>
     <th>Customer Name</th>
   </tr>
   <!-- dl4c(for) -->
   <tr>
     <td><a href="dl4v(S_CGI)?Session=dl4v(Session)

&Option=EXCUST&Action=Change&CUSTID=dl4a(url(CUSTID))">dl4a(CUSTID)</td></a>
     <td>dl4a(CUSTName)</td>
   </tr>
   <!-- dl4c(next) -->
<script language="Javascript">
<!--
var keynext="dl4v(keynext)";
if (keynext != "") {
document.write('<tr><td colspan=6 align=right><a
href="dl4v(S_CGI)?Session=dl4v(Session)&Option=EXCUST&Action=Search&Keyno=dl4v(
Keyno)&Keyvar=dl4v(url(Keyvar))&Keynext=dl4v(url(Keynext))">More</a></td></tr>'
);
}
//-->
</script>
</table>
</body>
</html>
```

HTML Code for custmnt.html

```
<html>
<body>
    <form method="POST" action="dl4v(S_CGI)">
       <input type="hidden" name="CUSTID" value="dl4v(CUSTID)">
       <input type="hidden" name="CUSTNewID" value="dl4v(CUSTNewID)">
       <table>
          <tr>
            <td>Customer ID</td>
            <td>dl4v(CUSTNewID)</td>
          </tr>
          <tr>
            <td>Customer Name</td>
            <td><input type="text" name="CUSTName" size="25"
                 value="dl4v(CUSTName)" maxlength="25"></td>
          </tr>
        </table>
        <p>
          <input type="submit" name="Action" value="Save">
          <input type="submit" name="Action" value="Delete">
        </p>
    </form>
</body>
</html>
```

dL4 code version for excust.dl4 program :
Note : Differences between Unibasic or dL4 :
In dL4 you can use a string array for D1$ and D2$ at lines 110,1140,1150,1310,1320.
In Unibasic you do not declare intrinsic calls and the format of the Call statements are different.note : In

```
1 Declare Intrinsic Sub DXOpen                          !dL4 only
2 Declare Intrinsic Sub DXGet
3 Declare Intrinsic Sub DXSet
4 Declare Intrinsic Sub DXClose

100 Dim A$[10],C$[10],K$[10],V$[10],E$[80],N$[25],3%
110 Dim D1$[40,10],D2$[40,25]                           !dL4 can be arrays D1$[40,10],D2$[40,25]
120 Call DXOpen()                                       !Open DynamicXport to get field values
130 If Err 0 Goto 4000
140 Open #1,"examples/Customers"
150 Call DXGet("Action",A$)                             !Read in value of Action field
160 If A$="Search" Gosub 1000 Else Gosub 2000           !Do appropriate routine
170 Call DXClose()                                      !Merge set fields to HTML page defined in
171 Rem                                                 !DynamicXport Option record
172 Rem                                                 !or Output=field
180 End

1000 Rem Search option
1010 Call DXGet("Keyvar",C$)                            !Get exact or partial value to search for
1020 Call DXGet("Keynext",K$)                           !Next key to search if continuation
1030 L=0                                                !Line counter
1040 V$=""                                              !Set beginning key
1050 If K$<>"" Let V$=K$                                !or continuation
1060 Search #1,2,1;V$,R,E                               !First key search
1070 Goto 1090
1080 Search #1,3,1;V$,R,E                               !Sequential search
1090 If E Goto 1300                                     !Goto set arrays and end
1100 If L>=40 Goto 1200                                 !Set Keyvar,Keynext and end

1110 Rem read name and load table
1120 Read #1,R;N$;
1130 If C$="" Goto 1140
1132 If Len(C$)<=Len(V$) If V$[1,Len(C$)]=C$  Goto 1140
1134 If Len(C$)<=Len(N$) If N$[1,Len(C$)]=C$ Goto 1140
1136 Goto 1080
1140 D1$[L]=V$                                          !dL4 can be string array D1$[L]=V$
1150 D2$[L]=N$                                          !dL4 can be string array D2$[L]=N$
1160 L=L+1
1170 Goto 1080

1200 Rem end at 40 lines
1210 Call DXSet("Keyvar",C$)                            !Set Keyvar to preserve search value
1220 Call DXSet("Keynext",V$)                           !Set Keynext to next key to display

1300 Rem set arrays and end
1310 Call DXSet("CUSTID",D1$[],"L")                     !Set CUSTID list, dL4 can be array D1$[]
1320 Call DXSet("CUSTName",D2$[],"L")                   !Set CUSTName list. dL4  D2$[]
1330 Return

2000 Rem Maintenance options
2010 If A$="Save" Goto 2800                             !Go to Save (add or change record) routine
```

```
2020 Call DXGet("CUSTID",C$)                                    !Read in Custid field
2030 V$=C$
2040 Search #1,2,1;V$,R,E                                       !Search for Customer
2050 If V$<>C$ Let E=2
2060 If A$="Add" Goto 2500                                      !Go to Add routine
2070 If E Let E$="Customer not on file" \ Goto 3000             !Set error if not on file, go to error routine
2080 If A$="Change" Goto 2600                                   !Go to Change routine
2090 If A$="Delete" Goto 2700                                   !Go to Delete routine
2100 Stop                                                       !End of valid options

2500 Rem Add routine
2510 If E=0 Let E$="Customer already on file" \ Goto 3000       !Set error if on file, go to error routine
2520 Call DXSet("CUSTID","")                                    !Set existing Custid to null to indicate add
2530 Call DXSet("CUSTNewID",C$)                                 !Set new Custid for output
2540 Call DXSet("CUSTName","")                  !Set default fields
2550 Call DXSet("Output","examples/custmnt.html")!Set HTML file to merge to
2560 Return

2600 Rem Change routine
2610 Read #1,R;N$;                                              !Read record to populate form
2620 Call DXSet("CUSTID",C$)                                    !Set existing Custid to indicate change mode
2630 Call DXSet("CUSTNewID",C$)                                 !Set new Custid
2640 Call DXSet("CUSTName",N$)                                  !Set default fields
2650 Call DXSet("Output","examples/custmnt.html")               !Set HTML file to merge to
2660 Return

2700 Rem Delete routine
2710 Search #1,5,1;V$,R,E                                       !Delete key
2720 E=3
2730 Search #1,1,0;V$,R,E                                       !Release record #
2740 E$="Customer ",C$," has been deleted!"                     !Set msg
2750 Goto 3000

2800 Rem Save (add or change record) routine
2810 Call DXGet("CUSTNewID",C$)                                 !Read new ID
2820 Call DXGet("CUSTName",N$)                                  !Read new field values
2830 V$=C$
2840 Search #1,2,1;V$,R,E                                       !Check if on file
2850 If V$<>C$ Let E=2
2860 If E Goto 2900                                             !Go to add record
2870 Write #1,R;N$;                                             !Change record
2880 E$="Customer ",C$," has been changed"                      !Set  msg
2890 Goto 3000
2900 Rem add record
2910 V$=C$
2920 E=2
2930 Search #1,1,0;V$,R,E                                       !Get free record #
2940 Write #1,R;N$;                                             !Write record
2950 Search #1,4,1;V$,R,E                                       !Add key
2960 E$="Customer ",C$," has been added"                        !Set msg
2970 Goto  3000

3000 Rem message routine
3010 Call DXSet("S_Msg",E$)                                     !Set standard message variable
3020 Call DXSet("Output","dxstd/msg.html")                      !Set HTML file to merge to
3030 Return
```

```
4000 Rem error message
4010 E$ = Msc$(2) + " at " + Spc(10)
4020 Gosub 3000
4030 Call DXClose()
4040 End
```

Unibasic code version for excust.ub program :
Note : Differences between Unibasic or dL4 :
In dL4 you can use a string array for D1$ and D2$ at lines 110,1140,1150,1310,1320.
In Unibasic you do not declare intrinsic calls and the format of the Call statements are different.

```
100 Dim A$[10],C$[10],K$[10],V$[10],E$[80],N$[25],3%
110 Dim D1$[440],D2$[1040]                          !dL4 can be arrays D1$[40,10],D2$[40,25]
120 Call $DXOpen                                    !Open DynamicXport to get field values
130 If Err 0 Goto 4000
140 Open #1,"examples/Customers"
150 Call $DXGet,"Action",A$                         !Read in value of Action field
160 If A$="Search" Gosub 1000 Else Gosub 2000       !Do appropriate routine
170 Call $DXClose                                   !Merge set fields to HTML page defined in
171 Rem                                             !DynamicXport Option record
172 Rem                                             !or Output=field
180 End

1000 Rem Search option
1010 Call $DXGet,"Keyvar",C$                        !Get exact or partial value to search for
1020 Call $DXGet,"Keynext",K$                       !Next key to search if continuation
1030 L=0                                            !Line counter
1032 L1=1                                           !D1$ length counter
1034 L2=1                                           !D2$ length counter
1040 V$=""                                          !Set beginning key
1050 If K$<>"" Let V$=K$                            !or continuation
1060 Search #1,2,1;V$,R,E                           !First key search
1070 Goto 1090
1080 Search #1,3,1;V$,R,E                           !Sequential search
1090 If E Goto 1300                                 !Goto set arrays and end
1100 If L>=40 Goto 1200                             !Set Keyvar,Keynext and end

1110 Rem read name and load table
1120 Read #1,R;N$;
1130 If C$="" Goto 1140
1132 If Len(C$)<=Len(V$) If V$[1,Len(C$)]=C$  Goto 1140
1134 If Len(C$)<=Len(N$) If N$[1,Len(C$)]=C$ Goto 1140
1136 Goto 1080
1140 D1$[L1]=V$                                     !dL4 can be string array D1$[L]=V$
1145 L1=L1+Len(V$)+1
1150 D2$[L2]=N$                                     !dL4 can be string array D2$[L]=N$
1155 L2=L2+Len(N$)+1
1160 L=L+1
1170 Goto 1080

1200 Rem end at 40 lines
1210 Call $DXSet,"Keyvar",C$                        !Set Keyvar to preserve search value
1220 Call $DXSet,"Keynext",V$                       !Set Keynext to next key to display

1300 Rem set arrays and end
1302 D1$[L1]=""                                     !Set final terminator
1304 D2$[L2]=""                                     !Set final terminator
1310 Call $DXSet,"CUSTID",D1$,"L"                   !Set CUSTID list, dL4 can be array D1$[]
1320 Call $DXSet,"CUSTName",D2$,"L"                 !Set CUSTName list. dL4  D2$[]
1330 Return
```

```
2000 Rem Maintenance options
2010 If A$="Save" Goto 2800                                  !Go to Save (add or change record) routine
2020 Call $DXGet,"CUSTID",C$                                 !Read in Custid field
2030 V$=C$
2040 Search #1,2,1;V$,R,E                                    !Search for Customer
2050 If V$<>C$ Let E=2
2060 If A$="Add" Goto 2500                                   !Go to Add routine
2070 If E Let E$="Customer not on file" \ Goto 3000          !Set error if not on file, go to error routine
2080 If A$="Change" Goto 2600                                !Go to Change routine
2090 If A$="Delete" Goto 2700                                !Go to Delete routine
2100 Stop                                                    !End of valid options

2500 Rem Add routine
2510 If E=0 Let E$="Customer already on file" \ Goto 3000    !Set error if on file, go to error routine
2520 Call $DXSet,"CUSTID",""                                 !Set existing Custid to null to indicate add
2530 Call $DXSet,"CUSTNewID",C$                              !Set new Custid for output
2540 Call $DXSet,"CUSTName",""                               !Set default fields
2550 Call $DXSet,"Output","examples/custmnt.html"            !Set HTML file to merge to
2560 Return

2600 Rem Change routine
2610 Read #1,R;N$;                                           !Read record to populate form
2620 Call $DXSet,"CUSTID",C$                                 !Set existing Custid to indicate change mode
2630 Call $DXSet,"CUSTNewID",C$                              !Set new Custid
2640 Call $DXSet,"CUSTName",N$                               !Set default fields
2650 Call $DXSet,"Output","examples/custmnt.html"            !Set HTML file to merge to
2660 Return

2700 Rem Delete routine
2710 Search #1,5,1;V$,R,E                                    !Delete key
2720 E=3
2730 Search #1,1,0;V$,R,E                                    !Release record #
2740 E$="Customer ",C$," has been deleted!"                  !Set msg
2750 Goto 3000

2800 Rem Save (add or change record) routine
2810 Call $DXGet,"CUSTNewID",C$                              !Read new ID
2820 Call $DXGet,"CUSTName",N$                               !Read new field values
2830 V$=C$
2840 Search #1,2,1;V$,R,E                                    !Check if on file
2850 If V$<>C$ Let E=2
2860 If E Goto 2900                                          !Go to add record
2870 Write #1,R;N$;                                          !Change record
2880 E$="Customer ",C$," has been changed"                   !Set  msg
2890 Goto 3000
2900 Rem add record
2910 V$=C$
2920 E=2
2930 Search #1,1,0;V$,R,E                                    !Get free record #
2940 Write #1,R;N$;                                          !Write record
2950 Search #1,4,1;V$,R,E                                    !Add key
2960 E$="Customer ",C$," has been added"                     !Set msg
2970 Goto  3000

3000 Rem message routine
3010 Call $DXSet,"S_Msg",E$                                  !Set standard message variable
```

```
3020 Call $DXSet,"Output","dxstd/msg.html"          !Set HTML file to merge to
3030 Return

4000 Rem error message
4010 E$ = Spc(8) , " at " , Spc(10)
4020 Gosub 3000
4030 Call $DXClose
4040 End
```

Wholesale Distribution Demo

A complete product search, order entry and order inquiry demo is included with dynamicXport.
All html and dL4 source code is provided and is a good source of coding examples of typical dynamic web development methods.

The Wholesale Distribution Demo is often referred to as DODEMO.

The html code for DODEMO can be found in the dx/examples directory and are all prefaced by the letters 'do'. Images used by the demo reside in dx/examples/images. Sample Product images reside in dx/examples/prodimg and sample product datasheet PDF's reside in dx/examples/prodspec.
These images initially reflect items for sale by an Electronics Distributor but can be replaced to customize the demo to your marketplace.

The dL4 application program for DODEMO is located in the examples/ directory on the application server.
The program name is dodemo.dl4 and the source is dodemo.txt.
The program name to rebuild the database is docreate.dl4. This program can be modified to customize the Product items used in the demo.

You will be prompted during installation if you wish to install the distribution demo.
If you install the DODEMO the data files will be built and the DO options added to the dynamicXport Options file and to the administrator's menu options. These Options can be removed later if desired.

Request Login/Forgot Password Options

Example html, Unibasic and dL4 programs are included to provide examples of the code necessary to provide web users with the functionality to request a login or request a new login (forgot password).

This functionality is not included in the standard dynamicXport capabilities because before automatically providing a login to the system a new user request would typically need to be verified against data stored in the custom application. For example, a client browsing the web can request a login to your application, but they would need to provide an Account #, phone number, social security number, D&B number, etc or combination thereof to verify the request is legitimate by comparing their inputs to your application database.

The html file for Request a Login is dx/examples/newuser.html.
The html file for Forgot Password is dx/examples/forgotpd.html.

The dL4 application program for both functions is called exnewuser.dl4 in the examples/ directory on the application server.
A Unibasic version of the program is called exnewuser.ub in the examples/ directory.

The html files and application program should be copied and then modified to meet your particular application's needs.

Forgot Password functionality is basically implemented by duplicating the Request a Login process and then deleting the old login.

The application program uses the DXBridge access to add and delete users in the dynamicXport Users database. A dynamicXport login and password, which has administration rights, must be coded into the application to facilitate using the bridge access.

Appendix E Credit Card Module

Module Definition

A module is included in the dynamicXport product to interface with Verisign's Payflow standard Java SDK for processing of Credit Card transactions over the internet.

All components of the Credit Card processing are implemented on the Application Server.
So although you can collect credit card payment information from a web page, dynamicXport does not store or process any credit card information on the more public web server.
Also, credit card information does not necessarily have to come from a web page since all credit card data is retained in your internal application and passed to the Verisign API for processing by your custom application on the Application Server.

The following files will be installed in the dynamicXport directory to support credit card processing :

dxverisignpayflow.dl4	dynamicXport credit card API wrapper program to CALL or run to process a transaction
payflowcfg.txt	dynamicXport credit card interface configuration file contains general configuration information for dynamicXport credit card processing with Verisign's Payflow SDK

The following types of processing are currently supported :

A	Card charge authorization only (Obtain approval to charge the card, reserve the right to charge the card, but does not actually charge the card. Typically done when an order is received or prior to shipping an order to insure the card can be charged.) (was mauthonly using CyberCash interface)
D	Capture of card charge only (Actually charge the card, must be preceeded by an approved authorization) (was postauth using CyberCash interface)
S	Authorize and charge the card in one process (Will obtain authorization to charge the card and if authorized will charge the card in one transaction. Typically done when charging for services or soft goods.) (was mauthcapture using CyberCash interface)
C	Credit, returns the specified amount to the account holder.
I	Inquiry transaction to check the result and status of a transaction.

Before processing credit cards the following must be done :

1. Obtain a Merchant key from Verisign and update the payflowcfg.txt file
    a. Verisign has two main web site addresses to go to.
    http://www.verisign.com/products/payment.html is for obtaining information on the Payflow Product and for signing up and creating an account.

Upon completion of registration, Verisign will provide you with a Partner ID, User ID (login) and Password.

https://manager.verisign.com is for processing and viewing transactions.

b. Costs will be incurred when a)obtaining a Merchant Account from a bank, if you do not already have one, b)monthly charges from Verisign for their processing service and c)charges from your bank for credit card transactions. An example of typical fees (may vary) :

Obtain Merchant Account from bank	$250 one-time charge
Verisign Setup	$250 setup fee
Verisign Monthly	$60 for under 1000 transactions
Bank fees	% of transaction

c. Make note of your Verisign Partner ID, User name and Password.
d. Edit the payflowcfg.txt file and change the S_USER, S_VENDOR, S_PARTNER, S_PWD and S_PAYFLOW fields.

You can create multiple payflowcfg.txt files with different login field values in order to process credit card transactions for different companies.
The configuration filename to use is specified by setting the S_CC_Config field in the application. If S_CC_Config is not set payflowcfg.txt will be used by default.

2. Download and install the Verisign Java API.

a. Log into the Verisign manager site, click download and download the Payflow Pro Pure Java SDK. Unzip and untar the downloaded file (pfpro_java.tar.gz). You may also want to download the Payflow Pro Developers Guide.
b. The Java 2 SDK must be downloaded and installed from the appropriate vendor (ie SCO)
c. Test the Java and Verisign Java API by editing and running the javatest.bat file

2. Edit the payflowcfg.txt file

The majority of the payflowcfg.txt file consists of mapping Verisign field names to your own, more reasonable field names. This is done with value pairs of Verisign field names = custom field names. If no custom field name is specified then the field name will remain the Verisign field name.

The payflowcfg.txt file must be located in the same directory as the dxverisignpayflow API.
Required fields for the common transaction types are :
Type S is sale, requires type,tender,acct,expdate,amt Other fields such as invoice number and comments are optional.
Type A is authorize only, same fields as sale type transactions.
Type D is charge after authorize, requires type and origid (equal to pnref returned from an A transaction).

3. Develop your custom application to call the dynamicXport API.

Credit Card processing is done by accessing the dxverisignpayflow.dl4 API program to talk to Verisign, similar to using the dxbridge.dl4 interface to talk to dynamicXport.
To process a credit card transaction the custom application must set the required and any desired optional field values using the Call DXSet statement. The TRXTYPE field must always be set to indicate the type of transaction being requested. After setting all the values, the dxverisignpayflow.dl4 API is called to do the processing. Upon return the Call DXGet statement is used to read the status and any other result field needed for the application.

Here is a dL4 example to perform a Sales transaction :

```
!Perform a credit card transaction routine

!The first Call DXset is to set the CC Configuration file to use for this transaction
!this is only necessary if you are processing with more than one merchant account and wish to override the
!default configuration file
!Call DXSet("S_CC_Config","/usr/xport/payflowcfg.txt")

!The next Call DXSet specifies the type of transaction
!S is sale, requires type,tender,acct,expdate,amt   Other fields optional
!A is authorize only, same fields as sale
!D is charge after authorize, requires type,origid (equal to pnref returned from A transaction)
Call DXSet("TRXTYPE","S")
!The next Call DXSet  specifies the tender, C=credit card
Call DXSet("tender","C")

!now set required and any desired optional fields for the type of transaction
!note that fieldnames are customizable and mapped in the payflowcfg.txt file to a Verisign fieldname
Call DXSet("invnum",ORDER$)                    !invoice# or order #
Call DXSet("comment1",ORDER$)                  !invoice# or order #
Call DXSet("acct",CCNUMBER$)                   !credit card #
Call DXSet("expdate",CCEXPIRES$)               !expire date MMYY
DOLLAR$=DOLLAR
Call DXSet("amt",DOLLAR$)

!now call the dxverisignpayflow.dl4 API to read the set variables and process the transaction
Call "/usr/xport/dxverisignpayflow.dl4"


!now get the status and any other desired fields
!note again that fieldnames are mapped in the payflowcfg.txt file
Call DXGet("pnref",PNREF$)                     !Verisign reference ID, length 12
Call DXGet("result",RESULT$)                   !number +/-9999
Call DXGet("respmsg",RESPMSG$)                 !variable length message, typically<40char.

!If the result is not equal to "0" the transaction failed. More details may be in the respmg field
If RESULT$<>"0"    --------

!Get other desired fields for your application
Call DXGet("authcode",AUTHCODE$)               !bank authcode, length 6

RESULT$="authorization code=",AUTHCODE$,"/pnref=",PNREF$

!that's it
```

Here is a Unibasic example to perform an Sales transaction :

```
1000 REM Perform a credit card transaction routine

1010 REM The first Call DXset is to set the CC Configuration file to use for this transaction
1020 REM this is only necessary if you are processing with more than one merchant account and wish to
1030 REM override the default configuration file
1040 Call $DXSet,"S_CC_Config","/usr/xport/payflowcfg.txt"

1050 REM The next Call DXSet specifies the type of transaction
1060 Call $DXSet,"TRXTYPE","S"
```

```
1062 REM The next Call DXSet  specifies the tender, C=credit card
1064 Call $DXSet,“tender”,”C”


1070 REM now set required and any desired optional fields for the type of transaction
1080 REM note that fieldnames are customizable and mapped in the payflowcfg.txt file to a Verisign
1090 REM fieldname
1100 Call $DXSet,“invnum”,ORDER$              !invoice# or order #
1110 Call $DXSet,”comment1”,ORDER$            !invoice# or order #
1120 Call $DXSet,“acct”,CCNUMBER$             !credit card #
1130 Call $DXSet,“expdate”,CCEXPIRES$         !expire date MMYY
1140 DOLLAR$=DOLLAR
1150 Call $DXSet,“amt”,DOLLAR$

1160 REM now call the dxverisignpayflow.dl4 API to read the set variables and process the transaction
1170 REM Unibasic needs to put set fields into a temporary input text file and define a
1180 REM temporary output text file to receive results, then run the bridge with a SYSTEM statement
1190 FILEIN$=”DXCCIN”,STR(SPC(6)),”!”         !Bridge input filename
1200 FILEOUT$=”DXCCOUT”,STR(SPC(6)),”!”       !Bridge output filename
1210 CALL $DXClose,FILEIN$                    !Save set fields to file
1220 S$=”/usr/bin/run –X /usr/xport/dxverisignpayflow.dl4 “,FILEIN$,” “,FILEOUT$
1230 SYSTEM(S$)

1240 REM now open output file to get results
1250 CALL $DXOpen,FILEOUT$

1260 REM now get the status and any other desired fields
1270 REM note again that fieldnames are mapped in the payflowcfg.txt file
1280 Call DXGet,“pnref”,PNREF$                !Verisign reference ID, length 12
1290 Call DXGet,“result”,RESULT$              !number +/-9999
1300 Call DXGet,“respmsg”,RESPMSG$            !variable length message, typically<40char.

1310 REM If the result is not equal to “0” the transaction failed. More details may be in the respmg field
1320 If RESULT$<>”0”    --------

1330 REM Get other desired fields for your application
1340 Call $DXGet,“authcode”,AUTHCODE$         !bank authcode, length 6
1370 RESULT$=”authorization code=”,AUTHCODE$,”/pnref=”,PNREF$
```

4. Other considerations

a. You may need to change your PATH environment variable to include the directory of the dynamicXport Credit Card API files. i.e. /usr/xport

5. Process test transactions

Prior to going ‘live’ test transactions should be run through your custom application, through the API to Verisign.

Upon completion of the API Call the STATUS field should be “0”.

When you are ready to process live transactions, modify the payflowcfg.txt and comment the test S_HostAddress and activate the regular S_HostAddress.