DYALOG

The tool of thought for expert programming

Dyalog™ for Windows

Workspace Transfer

Version 12.0.3

Dyalog Limited

South Barn
Minchens Court
Minchens Lane
Bramley
Hampshire
RG26 5BH
United Kingdom

tel: +44 (0)1256 830030
fax: +44 (0)1256 830031
email: support@dyalog.com
http://www.dyalog.com

Version 12.0.3

Revised August 2008

Contents

Contents iii

Workspace Transfer 1
Introduction 1
General Techniques 2
DWSOUT Workspace 4
DWSIN Workspace 7
Transferring Dyalog APL Workspaces 9
Importing APL*PLUS Workspaces 10
Importing IBM VS APL Workspaces 11
Importing IBM APL2 Workspaces 17

Workspace Transfer

Introduction

It is often necessary to either transfer Dyalog APL workspaces from one machine to another, or to transfer workspaces created by another version of APL to Dyalog APL.

Since the internal structure of an APL workspace is dependent upon the architecture of the machine on which it was created, and the version of APL that was used to create it, it is not possible to transfer a workspace directly. It must first be transformed into a format that is common to both source and target environments. This can then be transferred, and used to create a workspace on the target machine.

The following sections describe the steps that are involved in moving workspaces between machines and different versions of APL. APL component files are transferred in a similar way to workspaces; each component is read into an APL variable which is then transferred.

The subsequent sections describe specific examples of this process. The basic technique is the same in most cases. Doubtless better techniques can be used, but the following examples are useful as a basis from which to work.

The problems of conversion from one version of APL to Dyalog APL are not discussed in this section. Nor are the procedures for transferring from Dyalog APL to other APLs. However, using the techniques discussed in this section, it is a simple (if tedious) matter to produce software that will transfer in the opposite sections.

Where possible, the tools required to transfer to Dyalog APL are supplied with the product; where this is impractical, listings of the appropriate software is given.

General Techniques

Create a Script File

If we can produce a text file on the source machine that contains all of the statements necessary to recreate the workspace, we could execute each line of this file on the target machine using the Dyalog APL, thereby recreating the workspace in the correct format.

How do we transform a workspace into a text format? Functions are easy; we can just produce a listing of them. Variables are more difficult; we must produce lines of text that when executed, recreate the variables with the correct contents, type, shape and depth.

Any APL programmer would be capable of writing the simple APL system required to perform both of these tasks; however, a workspace called `DWSOUT` (Dyalog WorkSpace OUT) is supplied for certain versions of APL (Dyalog APL, APL*PLUS), and where the workspace cannot be supplied (VSAPL), appropriate listings are given.

Transfer to Target Machine

Once this file has been created, it should be transferred from the source to the target machine. This transfer may be done using diskettes or tapes (first ensure that the source media can be read by the target machine), or the file may be sent across some kind of network using file transfer software.

However the transfer is effected, the file should arrive on the target machine EXACTLY as it left the source machine; i.e. the transfer mechanism should not perform any conversion or translation. Note that the file produced by `DWSOUT` is a raw text file, and may contain any characters in the range 0 to 255; some network transfer packages react badly to characters in the range 0 to 32 unless you ask that the file is transferred in transparent mode.

Create Dyalog APL Workspace

Each line of the file must read and executed by Dyalog APL. Again, it would be a simple task for any APL programmer to write the required functions; however, a workspace called `DWSIN` (Dyalog WorkSpace IN) is supplied which can process files produced by Dyalog APL, APL*PLUS and VSAPL. It could easily be amended to cope with files produced by other APLs.

Exception

The exception to the above technique is the transfer of IBM APL2 workspaces. APL2 provides the user with a system command `)OUT` which produces a text file representing the workspace. This file can be transferred to the target machine, but must then be decoded in a special manner. See the relevant section for details.

DWSOUT Workspace

Introduction

Since different versions of APL have different mechanisms for creating and appending to native files, each version of APL requires a different version of the workspace `DWSOUT`. Versions currently exist for transferring from Dyalog APL, APL*PLUS and VSAPL. However, it would be a fairly simple matter for any of these workspaces to be amended to suit any version of APL.

Each workspace `DWSOUT` contains a function `DWSOUT` that is effectively a very simple workspace lister, which lists the contents of variables as well as listing functions. Where appropriate, a function `DCFOUT` is also supplied; this lists the contents of component files, by reading each component into a variable, and listing the variable using the same techniques as `DWSOUT`.

Each version of `DWSOUT` is essentially the same. Hence, the description of the use of `DWSOUT` given below is valid for all versions. The particular restrictions of each version are discussed in the relevant sections.

Transferring Workspaces

The function `DWSOUT` is used to transfer entire workspaces, or named objects from a workspace. The full syntax of the function call is:

WS Out `{names} ΔΔDWSOUT wsname`

`wsname` is a simple character vector containing the name under which the workspace is to be `)SAVE`d on the target system. `names`, if present, is a character matrix containing the names of the objects to be transferred. The default is the entire workspace.

A file called **wsname.DXF** is created, and listings of the requested objects, system variables and constants, and some control statements are appended to the file.

Restrictions

The following restrictions apply to all versions of DWSOUT:

- The state indicator must be empty.
- The names of objects to be transferred may not begin with ⍙⍙.
- Locked functions cannot be transferred.
- Namespaces are not transferred
- []NULLS and composed functions will not be transferred

Example

Create a text file called NEWWS.DXF containing a complete listing of a workspace called MYWS.

```
      )LOAD MYWS
saved ....

      )FNS
FOO GOO HOO

      )VARS
A B C

      )COPY DWSOUT
saved ...

      ⍙⍙DWSOUT 'NEWWS'
Functions ...
FOO, GOO, HOO,
Variables ...
A, B, C,
System Variables ...
Finished
```

Transferring Component Files

Where applicable, the function DCFOUT is used to transfer entire component files. The full syntax of the function call is:

```
File Out      {options} ΔΔDCFOUT filename
```

`filename` is a simple character vector containing the name of the file to be created on the target system.

`options`, if present, is either a simple text vector containing the name of the file to be transferred, or a two element vector whose first element contains the name of the file, and whose second contains a passnumber. If `options` is omitted, the file name is taken from `filename` with a passnumber of 0.

A file called `filename.DXF` is created. Each component in the file is read and assigned to a variable; this variable is then transferred using the same techniques as DWSOUT. The access matrix is read and transferred in the same manner.

Example

Create a text file called `PJB.DXF` containing a complete listing of a component file called PJB.

```
      )LOAD DWSOUT
saved ...

      ΔΔDCFOUT 'PJB'

Components ...
1, 2, 3, 4, 5, 6, 7, 8, 9, 10
Finished
```

DWSIN Workspace

The workspace DWSIN contains a function DWSIN that can process script files produced by the DWSOUT workspaces appropriate for several different APLs.

DWSIN is used to read the text files produced by DWSOUT and DCFOUT. Each line of the file is read and executed, thus recreating the workspace or component file. The full syntax of the function call is:

```
WS In          {source} ΔΔDWSIN filename
```

`filename` is a simple character vector containing the name of the file to be processed. This file must have been produced by a previous call of `DWSOUT`. Note that the suffix `".DXF"` is automatically appended to the given file name.

`source`, if present, is a simple text vector containing the name of the source APL, taken from the set `'DYALOG'`, `'APLPLUS'`, `'STSCMF'` or `'VSAPL'`. If `source` is omitted, the default is `'DYALOG'`.

`DWSIN` reads the file, applying the relevant translation from the source APL to Dyalog APL, and executes each line, thus recreating the objects.

If an object cannot be recreated because of badly formed lines in the file, then that object is ignored and a warning message printed. The names of such objects are held in a variable `WontFix`.

Example

Create a Dyalog APL workspace from the file `NEWWS.DXF`, which was created by Dyalog APL on another machine.

```
      )LOAD DWSIN
saved ...

      ⎕⎕DWSIN 'NEWWS'
Processing script ...
Functions & Operators ...
FOO, GOO, HOO,
Variables ...
System variables ...
*****************************
**** Workspace name is NEWWS
**** REMEMBER TO )SAVE IT !!!
*****************************
```

Example

Create a Dyalog APL component file from the file `PJB.DXF`, which was created by APL*PLUS/PC.

```
      )LOAD DWSIN
saved ...

      'APLPLUS' ⎕⎕DWSIN 'PJB'
Processing script ...
1, 2, 3, 4, 5, 6, 7, 8, 9, 10
Finished.
```

Transferring Dyalog APL Workspaces

Dyalog APL workspaces and component files are binary compatible across machines with similar architectures. For example, workspaces created under Dyalog APL for DOS can be used immediately by Dyalog APL under Xenix, as long as both environments are running the same version of Dyalog APL. However, if you are not sure if the source and target machines are compatible, it is best to transfer workspaces using `DWSOUT` and `DWSIN`.

The `DWSOUT` workspace for Dyalog APL copes with nested arrays, defined functions and operators, assigned functions, `⎕OR` objects and component files. However, the following restrictions apply:

- Arrays containing the `⎕OR` of an assigned function cannot be transferred.
- Locked functions cannot be transferred. However, a matrix of the names of the locked functions is created and this transferred. This variable is called `LockedFns`.
- Functions such as `SUM←+/` cannot be transferred. A matrix of the names of such functions, together with a *best estimate* of their contents, is created in the target workspace. This variable is called `AssignedFns`.
- Any workspace name supplied to `DWSOUT` or file name supplied to `DCFOUT` must not exceed 10 characters.

Importing APL*PLUS Workspaces

The DWSOUT workspaces for both APL*PLUS/PC, APL*PLUS II and APL+Win can be found in the folder tools\aplplus folder below the main Dyalog program folder.

The following restrictions apply to their use:

- Functions containing references in their header line to APL*PLUS specific system variables cannot be fixed by Dyalog APL. It is recommended that such functions are edited on the source machine before transfer. The following system variables cause a problem if they occur in a header line:

```
⎕ALX, ⎕CURSOR, ⎕ELX, ⎕HNAMES, ⎕HTOPICS,
⎕KEYW, ⎕SA, ⎕SEG, ⎕WINDOW
```

- No attempt is made to convert APL*PLUS specific code to the Dyalog APL equivalent.
- Any workspace name supplied to DWSOUT or file name supplied to DCFOUT must not exceed 8 characters.
- All graphic characters are transferred, but to their single line equivalents.
- Any character not in Dyalog APL's character set is translated as ⎕AV[⎕IO+127].
- ⎕AV[⎕IO+255] (alternative space) is translated to ⎕AV[⎕IO+32] (space) before transfer.
- Component files are assumed to start at component one.

Importing IBM VS APL Workspaces

A listing of an appropriate DWSOUT is supplied. You must amend the functions ΔΔFILE_CREATE, ΔΔFILE_WRITE and ΔΔFILE_CLOSE to suit your particular operating system.

You need only use the leading characters ΔΔ for all function and variable names if you wish to avoid any name clashes.

This version of DWSOUT is subject to the following restrictions (which you can code around if you so wish):

- No error checking is performed.
- No system variables are transferred.
- All objects in the workspace are transferred; there is no option for transferring a subset.
- Functions are transferred in ⎕CR form, and trailing blanks are not removed.

```
ΔΔDOUBLE
========

     ∇ ΔΔA←ΔΔB ΔΔDOUBLE ΔΔC;ΔΔD;ΔΔE;ΔΔF;⎕IO
[1]   ⍝ REPLACE CHAR ΔΔB WITH (2⍴ΔΔB) IN STRING ΔΔC
[2]    ⎕IO←0
[3]    ΔΔF←(ΔΔB=ΔΔC)/⍳ΔΔD←⍴ΔΔC←,ΔΔC
[4]    ΔΔE←(ΔΔD+⍴ΔΔF)⍴1
[5]    ΔΔE[ΔΔF←ΔΔF+⍳⍴ΔΔF]←0
[6]    ΔΔA←ΔΔE\ΔΔC
[7]    ΔΔA[ΔΔF]←ΔΔB
     ∇


ΔΔDWSOUT
========

     ∇ ΔΔDWSOUT ΔΔFILENAME;⎕IO;⎕PP;⎕PW;ΔΔEOL;ΔΔNEWNAME
[1]   ⍝ PRODUCES A SCRIPT FROM AN IBM VSAPL WORKSPACE
[2]    ⎕IO←1
[3]    ⎕PP←16
[4]    ⎕PW←255
[5]   ⍝ DATA FILE NAME (MUST END IN .DXF ON TARGET MA
        CHINE)
[6]    ΔΔNEWNAME←(ΔΔFILENAME≠' ')/ΔΔFILENAME
[7]    ΔΔFILENAME←ΔΔNEWNAME,'.DXF'
[8]   ⍝ SET END OF LINE
[9]    ΔΔEOL←⎕AV[256]
[10]  ⍝ OPEN DATA SET
[11]   ΔΔFILE_CREATE
[12]  ⍝ SEND CONTROL STATEMENTS
[13]   ΔΔOUT')WSID←''',ΔΔNEWNAME,''''
[14]  ⍝ TRANSFER FUNCTIONS
[15]   ΔΔOUTFNS
[16]  ⍝ TRANSFER VARIABLES
[17]   ΔΔOUTVARS
[18]  ⍝ AND FINISH
[19]   ΔΔFILE_CLOSE
[20]   ΔΔTOSCREEN ⎕TC[2],'FINISHED'
     ∇

ΔΔFILE_CLOSE
============

     ∇ ΔΔFILE_CLOSE
[1]   ⍝ CLOSE DATA FILE ΔΔFILENAME - SYSTEM DEPENDENT
     ∇
```

```
ΔΔFILE_CREATE
=============

     ∇ ΔΔFILE_CREATE
[1]   ⍝ CREATE FILE ΔΔFILENAME - SYSTEM DEPENDENT
     ∇


ΔΔFILE_WRITE
============

     ∇ ΔΔFILE_WRITE ΔΔTEXT
[1]   ⍝ WRITE ΔΔTEXT TO FILE ΔΔFILENAME - SYSTEM DEPE
        NDENT
     ∇


ΔΔOUT
=====

     ∇ ΔΔOUT ΔΔLINE;ΔΔN;ΔΔR
[1]   ⍝ PASS LINE OF TEXT TO FILE, FINISHING WITH AN
        END OF LINE CHAR
[2]    ΔΔFILE_WRITE ΔΔLINE,ΔΔEOL
     ∇


ΔΔOUTFN
=======

     ∇ ΔΔOUTFN ΔΔNAME;ΔΔ
[1]   ⍝ OUTPUT FUNCTION AS A VARIABLE, THEN FIX
[2]    →ΔΔGO×⍳0≠1↑⍴ΔΔ←⎕CR ΔΔNAME
[3]    ΔΔTOSCREEN' (LOCKED)'
[4]    →0
[5]   ΔΔGO:ΔΔOUTVAR'ΔΔ'
[6]    ΔΔOUT'⎕FX ΔΔ'
     ∇
```

```
ΔΔOUTFNS
========

     ∇ ΔΔOUTFNS;ΔΔFNS;ΔΔI
[1]   ⍝ OUTPUT ALL NON-XFER FUNCTIONS
[2]    ΔΔTOSCREEN ⎕TC[2],'FUNCTIONS '
[3]    ΔΔFNS←(ΔΔFNS[;1 2]∨.≠'ΔΔ')⌿ΔΔFNS←⎕NL 3
[4]    ΔΔI←0
[5]   ΔΔNEXTI:→((1↑⍴ΔΔFNS)<ΔΔI←ΔΔI+1)/ΔΔEND
[6]    ΔΔTOSCREEN'... ',ΔΔFNS[ΔΔI;]
[7]    ΔΔOUT'⍝',ΔΔFNS[ΔΔI;]
[8]    ΔΔOUTFN ΔΔFNS[ΔΔI;]
[9]    →ΔΔNEXTI
[10]  ΔΔEND:
     ∇
```

```
ΔΔOUTVAR
========

     ∇ ΔΔOUTVAR ΔΔNAME;ΔΔSHAPE;ΔΔVALUE;ΔΔN;ΔΔM;ΔΔX;ΔΔ
       Q;ΔΔI
[1]    ⍝ SPLITS VARIABLES INTO MANAGEABLE CHUNKS
[2]    ΔΔVALUE←⍎ΔΔNAME
[3]    ΔΔSHAPE←⍴ΔΔVALUE
[4]    ΔΔVALUE←,ΔΔVALUE
[5]    ⍝ TREAT NUMERIC AND TEST DIFFERENTLY
[6]    →(ΔΔNUM,ΔΔTXT)[1+0≠1↑0⍴ΔΔVALUE]
[7]    ⍝ NUMERIC
[8]    ΔΔNUM:ΔΔOUT'ΔΔ←',(⍕×/ΔΔSHAPE),'⍴0'
[9]    ΔΔN←10
[10]   ΔΔI←0
[11]   ΔΔL1:→(0=⍴ΔΔVALUE)/ΔΔEXIT
[12]   ΔΔOUT'ΔΔ[',(⍕ΔΔI),'+⍳',(⍕ΔΔM),']←',⍕(ΔΔM←ΔΔN⌊
       ⍴ΔΔVALUE)↑ΔΔVALUE
[13]   ΔΔVALUE←ΔΔM↓ΔΔVALUE
[14]   ΔΔI←ΔΔI+ΔΔM
[15]   →ΔΔL1
[16]   ⍝ TEXT
[17]   ΔΔTXT:ΔΔOUT'ΔΔ←',(⍕×/ΔΔSHAPE),'⍴'' '''
[18]   ΔΔQ←''''
[19]   ΔΔN←60
[20]   ΔΔI←0
[21]   ΔΔL2:→(0=⍴ΔΔVALUE)/ΔΔEXIT
[22]   ΔΔX←(ΔΔM←(⍴ΔΔVALUE)[ΔΔN-+/ΔΔQ=ΔΔN↑ΔΔVALUE)↑ΔΔ
       VALUE
[23]   ΔΔOUT'ΔΔ[',(⍕ΔΔI),'+⍳',(⍕ΔΔM),']←''',(ΔΔQ ΔΔDO
       UBLE ΔΔX),ΔΔQ
[24]   ΔΔVALUE←ΔΔM↓ΔΔVALUE
[25]   ΔΔI←ΔΔI+ΔΔM
[26]   →ΔΔL2
[27]   ⍝ ISSUE CODE TO RESHAPE VARIABLE AND ASSIGN TO
       VARIABLE NAME
[28]   ΔΔEXIT:ΔΔOUT'ΔΔ←',((4×0≠⍴ΔΔSHAPE)↓'(⍳0)',⍕ΔΔSHA
       PE),'⍴ΔΔ'
[29]   ΔΔOUT ΔΔNAME,'←ΔΔ'
     ∇
```

```
ΔΔOUTVARS
=========

     ∇ ΔΔOUTVARS;ΔΔVARS;ΔΔI
[1]   ⍝ PRODUCES SCRIPT TO RECONSTITUTE VARIABLES
[2]    ΔΔTOSCREEN ⎕TC[2],'ΔΔVARIABLES '
[3]    ΔΔVARS←(ΔΔVARS[;⍳2]∨.≠'ΔΔ')⌿ΔΔVARS←⎕NL 2
[4]    ΔΔI←0
[5]   ΔΔL:→((1↑⍴ΔΔVARS)<ΔΔI←ΔΔI+1)/ΔΔEND
[6]    ΔΔTOSCREEN'... ',ΔΔVARS[ΔΔI;]
[7]    ΔΔOUT'⍝',ΔΔVARS[ΔΔI;]
[8]    ΔΔOUTVAR ΔΔVARS[ΔΔI;]
[9]    →ΔΔL
[10]  ΔΔEND:
     ∇


ΔΔTOSCREEN
==========

     ∇ ΔΔTOSCREEN ΔΔMSG
[1]   ⍝ DISPLAY MESSAGE AT TERMINAL
[2]    Δ←ΔΔMSG
     ∇
```

Importing IBM APL2 Workspaces

IBM APL2 has a system function `)OUT` which produces a text file that represents the workspace. This file must be transferred and decoded line by line on the target machine. This decoding is performed by workspaces `APL2IN` and `APL2PCIN`. **NOTE**: those workspaces are not supplied on the distribution disk but can be found on Dyalog's Website.

`APL2IN` is suitable only for importing workspaces from mainframe APL2; `APL2PCIN` performs the same task for workspaces exported from APL2/PC.

Create a script file

Load your workspace, and use `)OUT` to create a file.

Transfer to target machine

Move file to target machine; no translation should be performed.

Create Dyalog APL workspace

Use the workspace `APL2IN` or `APL2PCIN` to process the script file. This workspace contains a function called `⍙⍙APL2IN`, which takes as its right argument the name of the file that has been transferred from APL2. Once the file has been processed, you must save the active workspace.

Example

a) On mainframe:

```
        )LOAD MYWS
saved ...

        )OUT
```

b) Transfer file to target machine, with NO translation

c) On target machine:

```
        )LOAD APL2IN
saved ...

        ⍙⍙APL2IN 'filename'

        )SAVE newname
```

Importing APLX Workspaces

The procedure is the same as for APL2 workspaces (see above), except that you use the supplied workspace ATFIN instead.

Example

a) In APLX:

```
      )LOAD MYWS
saved ...

      )OUT
```

b) Transfer file to target machine, with NO translation

c) On target machine:

```
      )LOAD ATFIN
saved ...

      ΔΔatfin 'filename'

      )SAVE newname
```

Importing APL+Win Workspaces

The procedure is the same as for APL2 workspaces (see above). You use the same APL2IN workspace to do the import. APL+Win uses User Command]OUT to do the work. The APL2PCIN workspace also works for APL+Win so it can be used here.

Example

a) In APL+Win:

```
        )LOAD MYWS
saved ...

        ]OUT filename
```

b) Transfer file to target machine, with NO translation

c) On target machine:

```
        )LOAD APL2PCIN
saved ...

        ΔΔAPL2IN 'filename'

        )SAVE  newname
```