



The tool of thought for expert programming

Dyalog™ for Windows

Release Notes

Version: 13.2

Dyalog Limited

email: support@dyalog.com

<http://www.dyalog.com>

Dyalog is a trademark of Dyalog Limited

Copyright © 1982-2013 by Dyalog Limited

All rights reserved.

Version: 13.2

Revision: 22186

No part of this publication may be reproduced in any form by any means without the prior written permission of Dyalog Limited.

Dyalog Limited makes no representations or warranties with respect to the contents hereof and specifically disclaims any implied warranties of merchantability or fitness for any particular purpose. Dyalog Limited reserves the right to revise this publication without notification.

TRADEMARKS:

SQAPL is copyright of Insight Systems ApS.

UNIX is a registered trademark of The Open Group.

Windows, Windows Vista, Visual Basic and Excel are trademarks of Microsoft Corporation.

Oracle and Java are registered trademarks of Oracle and/or its affiliates.

Array Editor is copyright of davidliebtog.com

All other trademarks and copyrights are acknowledged.

Contents

Chapter 1: Introduction	1
Key Features	1
System Requirements	4
Array Editor	5
Performance Improvements	8
Selective Assignment	10
Memory Manager Statistics	10
Enhancements for Icon and Bitmap Objects	11
ListView Enhancements	12
New Button Styles	14
Change to Grid appearance (not XP)	16
Miscellaneous Enhancements	19
Announcements	22
Chapter 2: Language Reference Changes	25
Assignment (Selective):	26
Assignment (Indexed):	28
Memory Manager Statistics:	33
Chapter 3: Object Reference Changes	37
Native Look and Feel	38
ButtonEdit	40
ButtonsAcceptFocus	42
Cue	43
HeaderImageIndex	44
HeaderImageList	44
HighlightHeaders	45
HThumbDrag	45
Masked	46
PageSize	46
ReportBCol	47
ReportImageIndex	47
RowHiddenDepth	47
SelectionBorderWidth	48
SelectionColor	48
SelectionColorAlpha	48
ShowBalloonTip	49
ShowCueWhenFocused	50
VThumbDrag	51

Index **53**

Chapter 1:

Introduction

Key Features

Dyalog APL Version 13.2 provides the following new features, enhancements and changes:

General Enhancements and Improvements


- Performance Improvements
- Extensions to Selective Assignment
- Extensions to `2000⍒` (Memory Manager Statistics)
- `⌈FREAD` has been extended to allow you to read a vector of components in a single atomic operation.

New GUI Features

- New ButtonEdit Object. See ["ButtonEdit" on page 40](#)
- New CommandLink and Split Button Styles. See ["New Button Styles" on page 14](#)
- Cue Property for the Edit and ButtonEdit Objects
- ShowBalloonTip Method for the SysTrayItem Object
- HeaderImageList, HeaderImageIndex, ReportImageIndex, and ReportBCol Properties for the ListView Object. See ["ListView Enhancements" on page 12](#)
- SelectionColor, SelectionColorAlpha, SelectionBorderWidth, RowHiddenDepth and HighLightHeaders Properties for the Grid Object
- ButtonsAcceptFocus Property for the ToolControl Object
- PageSize Property for the Form, SubForm and Scroll Objects and HThumbDrag and VThumbDrag Events for the Form and SubForm Objects
- In previous Versions of Dyalog APL, a manifest file was required to enable *Native Look and Feel*. This is no longer needed. See ["Native Look and Feel" on page 38](#)
- Under all supported versions of Windows except XP, the colours of Grid lines and row and column headers now respects the user's choice of theme if

Native Look and Feel is enabled. See ["Change to Grid appearance \(not XP\)" on page 16](#)

Other

- Version 13.2 Unicode Edition includes the Array Editor which allows you to edit arbitrary arrays, including nested arrays. In the Unicode Edition the  icon in the Session now invokes the Array Editor rather than the numeric editor which was provided in previous Versions. The Classic Edition continues to use the numeric editor. See ["Array Editor" on page 5](#)
- Version 13.2 includes Conga 2.3 which provides support for Integrated Windows Authentication

Announcements

From Version 14.0 the following features will cease to be supported or will change:

- Support for Version 12.1 and Version 13.0. See ["Support for Version 12.1 and 13.0" on page 22](#)
- 32-bit component files. See ["Deprecation of 32-bit component files" on page 22](#)
- Auxiliary Processors. See ["Auxiliary Processors" on page 23](#)
- Default random number generator. See ["Random Number Generator" on page 23](#)

System Requirements

Microsoft Windows

Dyalog APL Version 13.2 supports versions of Windows from Windows XP up to and including Windows 8 and Windows Server 2012.

Dyalog APL Version 13.2 is not supported for versions of Windows prior to Windows XP, such as Windows 2000, Windows 98, Windows 95, Windows ME and Windows NT4.

Microsoft .Net Interface

Dyalog APL Version 13.2 .Net Interface requires Version 2.x or greater of the Microsoft .Net Framework. It does *not* operate with .Net Version 1.0.


Unix and Linux

For AIX, Version 13.2 requires AIX6.1 or higher, and a POWER5 chip or higher.

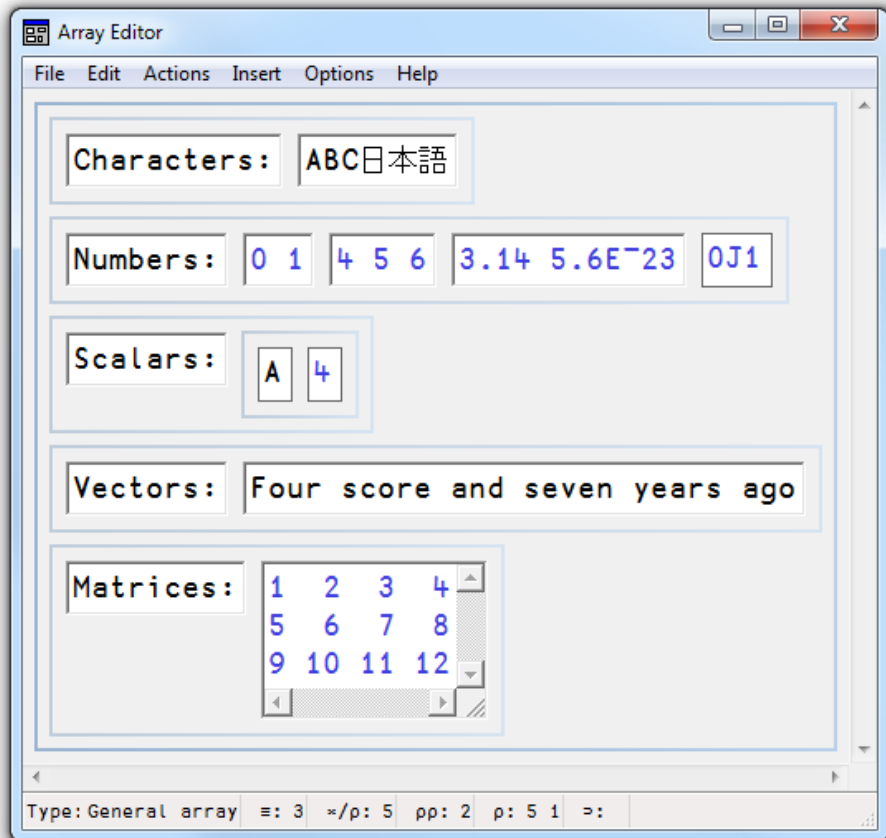
Version 13.2 is built on RedHat5, and runs on all recent distributions, including Ubuntu 12.04 and openSUSE 12.2. Contact Dyalog for other platforms.

Array Editor

The Array Editor¹ allows you to edit arbitrary arrays. It is invoked by either:

- Clicking the  icon in the Session toolbar when the mouse pointer is over the name of a suitable variable.
- Calling the User Command `]aedit`, specifying the name of a suitable variable as its argument.
- Calling it directly via `[NA`

The Array Editor draws data using a format that is similar to the output of the `DISPLAY` function. For example:



¹Array Editor Version 1 Release 1 © Copyright davidliebtag.com 2012, 2013

Documentation

Full documentation for the Array Editor, including a list of the keystrokes it uses, is available from the Help menu in the Array Editor's window.

Supported Arrays

The Array Editor supports arrays that consist solely of characters and/or numbers. You may not use it to edit an array that contains an object reference or a `⎕OR`.

Reject unsupported data

The way that the Arrays Editor reacts to unsupported arrays is determined by the value of the **Reject unsupported data** option which is accessed by the *Options/Reject unsupported data* menu item on the Array Editor menubar.

If this is set to true (the default), and you try to edit an array containing an object reference, the Array Editor will refuse the start and the system will generate an error message.

```
⎕SE.NumEd.numed: Unexpected error in array editor:
                    DOMAIN ERROR Argument contained data that is neither simple or
nested.
```

If this option is cleared, the Array editor will start but you will not be able to do anything. It is therefore advisable that you leave this option set.


Notes

- The Array Editor is supplied only with Unicode Editions of Dyalog APL/W. Please visit www.davidliebtag.com for details about availability and support for Classic Editions of Dyalog APL/W.
- Namespaces are not supported.
- Internal representations returned by `⎕OR` are not supported.
- Only one instance of the Array Editor may be executed at a time.
- All calls to interpreter primitives use a value of 3 for `⎕ML`.
- Negative numbers must be represented using high minus signs. For example, `¯3` not `-3`.

Implementation

The Array Editor is implemented by a DLL named `dlaedit.dll` (32-bit) or `dlaedit64.dll` (64-bit).

The DLL exports two functions: `DyalogEditArray` and

`DyalogEditArrayTitle`. The latter is used when you click the  icon in the Session toolbar (via the APL function `SE.NumEd.numed`) and by the User Command `J aedit`

Calling the Array Editor Directly

If you wish to use the Array Editor directly, you may do so as follows using `NA`¹.

For both `DyalogEditArray` and `DyalogEditArrayTitle` the first argument is the array to be edited, while the second argument is a place holder and should always be 0

For `DyalogEditArrayTitle` the 3rd argument is a character vector whose contents are displayed in the caption of the array editor window.

The result is the newly altered array.

Examples

```

NA'dlaedit.dll|DyalogEditArray <pp >pp'      A 32-bit
NA'dlaedit.dll|DyalogEditArrayTitle <pp >pp <0C2[]' A 32-bit

NA'dlaedit64.dll|DyalogEditArray <pp >pp'    A 64-bit
NA'dlaedit64.dll|DyalogEditArrayTitle <pp >pp <0C2[]' A 64-bit

New←DyalogEditArray Old 0
New←DyalogEditArrayTitle Old 0 Name

```

¹Note that these are not standard `NA` calls, but rather use an extension to `NA`, called *Direct Workspace Access*. Dyalog does not intend to make this feature generally available at present: if you are interested in this feature please contact sales@dyalog.com.

Performance Improvements

Reduction Operator

The performance of certain reductions on non-vector numeric arrays has been improved. $+/$, $\times/$, $\uparrow/$, and $\downarrow/$ on floating point numbers are faster by a factor of 6 while $+f$, $\times f$, $\uparrow f$, and $\downarrow f$ are faster by a factor of 2. In addition, $\uparrow/$ and $\downarrow/$ are faster by the same factors on other numeric types, and likewise $\uparrow f$ and $\downarrow f$.

Scan Operator

The time taken for $\leq\backslash$, $>\backslash$, $\geq\backslash$, $\checkmark\backslash$, and $\tilde{\times}\backslash$ has reduced from order n^2 to order n , where n is the length of the argument.

Miscellaneous

The performance of the following expressions, which mostly involve operations on Boolean arrays, has been improved. The degree of performance improvement is indicated in the column labelled *Factor*.

The meaning of the symbols used in the expressions is as follows:

n	integer scalar
i	integer array
b	Boolean array
x	array

Expression	Factor	Comments
$n i$	19-34	When n is a power of 2.
$2 \perp b$	17-21	
ϕb and $\ominus b$	2-93	
$(n \rho 2) \uparrow i$	19-21	
$x[b; \dots;]$	2-17	
ιn	1-2	
ϕx	4-230	
$?i \rho 2$	2-30	when $\square IO \leftarrow 0$
$b / \iota \rho b$ and $b / \iota n$	3.5	

Search and Replace

`⌘R` and `⌘S` operate significantly faster than before for simple searches – essentially, those without regular expressions, function transformations, options other than *Mode* or nested input documents.

Pick in Selective Assignment

The performance Pick (`=>`) in selective assignment expressions is substantially faster than before.

Stranding of Arrays

The performance of the stranding of named arrays, as in expressions such as:

```
A B C D E F
```

has been improved. The time taken has been reduced from being an order of n^2 to an order of n , where n is the number of names in the strand.

Account Name

The value of `⌘AN` is now cached in the workspace rather than being obtained dynamically every time it is referenced. Since `⌘AN` is used internally whenever a function is fixed applications which fix a very large number of functions will benefit from this caching.

Selective Assignment

Selective Assignment has been extended so that:

The Each Operator may now be used in selection expressions.

The syntax of selective assignment expressions is extended from

```
(EXP X)←Y
```

to

```
(EXP X)[I]←Y  a Case 1  
(EXP X[I])←Y  a Case 2
```

where:

- **EXP** is a selection expression using primitive functions from the table ["Functions for Selective Assignment" on page 26](#)
- **X** is the name of a variable
- **Y** is an array of items that replace the selected items in **X**.
- **[I]** is square-bracket (simple, choose, reach) indexing.

These enhancements bring Dyalog into line with mainframe IBM®APL2®.

For further details and examples, see ["Assignment \(Selective\):" on page 26](#)

Memory Manager Statistics

Function **2000I** (Memory Manager Statistics) has been extended to provide:

1. The ability to determine an application's peak memory usage so that its minimum value for **MAXWS** can be determined.
2. The ability to set an upper limit on the amount of workspace actually committed by the Operating System to prevent code which grabs as much workspace as it can from skewing the peak usage result.
3. The ability to set a lower limit on the workspace allocation to avoid repeatedly committing and releasing memory to the Operating System when memory usage is fluctuating.

For further details and examples, see ["Memory Manager Statistics:" on page 33](#)

Enhancements for Icon and Bitmap Objects

Bitmap and Icon objects can now be created from files of type GIF, JPG and PNG in addition to the file types previously supported (BMP and ICO respectively). However, they may only be written (using the FileWrite method) to BMP and ICO files as before.

The size of the image displayed by an Icon object was previously determined by its Style property which was either 'Small' (16x16) or 'Large' (32x32). The image size is now overridden by the value of the Size property which may contain any appropriate value. Note that a ICO file can contain images of multiple sizes.

The Icon object now supports 32-bit icons (24-bit images with an 8-bit alpha channel) using the CBits property instead of the Bits, CMap and Mask properties. The Bitmap object also supports the same type of transparent images.

The Masked property of the ImageList object has been extended to support high-colour transparent images.

ListView Enhancements

The ListView object may now display icons against column titles and items in Report View.

In Report View, the HeaderImageList property identifies the ImageList object containing the icons, and the HeaderImageIndex property specifies which icons from this ImageList are to be displayed in which column headings.

In the example shown below, the ImageList identified by HeaderImageList contains 4 icons for the 4 different times of day shown in the column headings and HeaderImageIndex is `[-2 0 1 2 3 4]`. The value `-2` specifies the standard *sort down* symbol.

In Report View, the ReportImageIndex property now overrides the ImageIndex property and additionally specifies which icons, from the ImageList specified by ImageListObj, are to be displayed against each item in the matrix specified by ReportInfo.

ReportImageIndex is a matrix whose first column specifies the indices of the icons to be displayed against the Items of the ListView, and whose subsequent columns specify the indices of the icons to be displayed against the elements of ReportInfo.

i.e. if non-scalar, $(\rho\text{ReportImageIndex}) \leftrightarrow (0 \ 1 + \rho\text{ReportInfo})$

In the example shown below, the ImageList associated with ImageListObj contains icons for each country's flag, followed by icons for the different types of weather.

`ReportImageIndex[;1]` is `1\rhoCOUNTRIES`

`ReportImageIndex[;2]` is 0

`ReportImageIndex[3 4 5 6;]` is a set of integers (`>1\rhoCOUNTRIES`) which index the weather icons

Country	Capital	08:00	14:00	20:00	02:00
Australia	Canberra	Few Clouds	Very Cloudy	Thunder Storm	Scattere...
Brazil	Brasilia	Cloudy	Very Cloudy	Clear	Light Rain
Canada	Ottawa	Scattered Cloud	Few Clouds	Light Rain	Cloudy
Denmark	Copenhagen	Cloudy	Very Cloudy	Cloudy	Scattere...
Finland	Helsinki	Scattered Cloud	Clear	Scattered Cloud	Scattere...
France	Paris	Rain	Few Clouds	Very Cloudy	Light Rain
Germany	Berlin	Clear	Clear	Clear	Light Rain
Ireland	Dublin	Cloudy	Thunder Storm	Very Cloudy	Light Rain
Italy	Rome	Thunder Storm	Light Rain	Few Clouds	Light Rain
Japan	Tokyo	Thunder Storm	Scattered Cloud	Light Rain	Thunder ...
United Kingdom	London	Clear	Scattered Cloud	Thunder Storm	Rain
United States	Washington	Clear	Few Clouds	Cloudy	Rain

The ReportBCol property optionally specifies the background colours for each element in Report View. Its first column refers to the Items themselves, and subsequent columns to the elements of ReportInfo.

i.e. if non-scalar, ($\rho\text{ReportBCol}$) \leftrightarrow (0 1+ $\rho\text{ReportInfo}$)

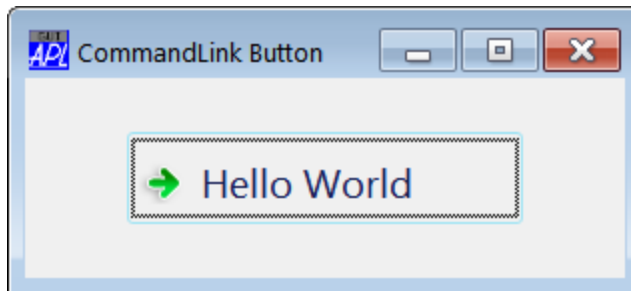
New Button Styles

There are two new Styles for the Button Object, namely 'CommandLink' and 'Split'. These apply only to Windows Vista and later and require Native Look and Feel. See "[Native Look and Feel](#)" on page 38. Otherwise the use of these Styles will produce a Button with Style 'Push'.

CommandLink Style

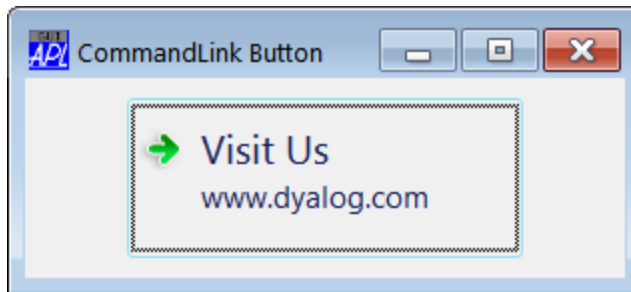
The CommandLink button has an icon displayed to the left of its Caption.

```
'F'[]WC'Form' 'CommandLink Button'  
F.clb'[]WC'Button' 'Visit Us'('Style' 'CommandLink')
```



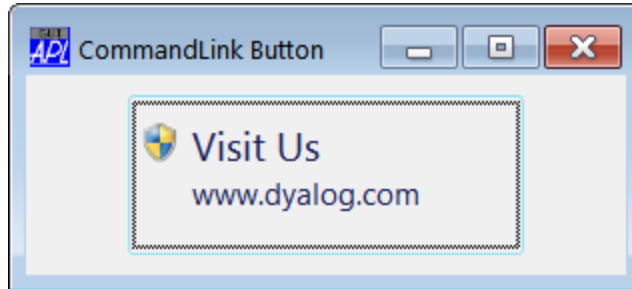
In addition to the caption, additional text may be defined by its Note property. If provided, this is displayed below the Caption.

```
F.clb.Size←80 200  
F.clb.Note←'www.dyalog.com'
```



The `Elevated` property is a Boolean scalar which when set to 1, changes the icon on the `CommandLink` button. This is intended to convey to the user that the action associated with the Button requires *Elevation*, a feature of *User Account Control* in Windows 7.

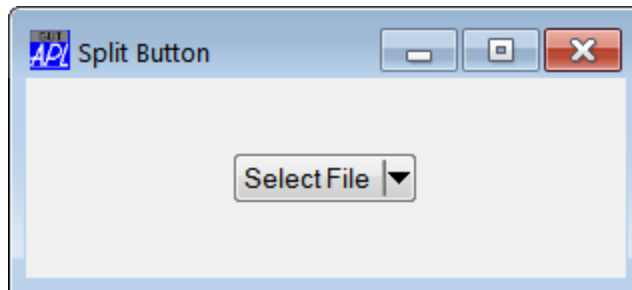
```
F.clb.Elevated=1
```



Split Style

The `Split Button` has a drop-down button, similar to that provided by a `Combo` object. The user can interact with the Button by clicking it, which generates a `Select` Event as before, or by clicking the drop-down, which generates a `DropDown` Event. It is up to the programmer to handle the `DropDown` Event as appropriate.

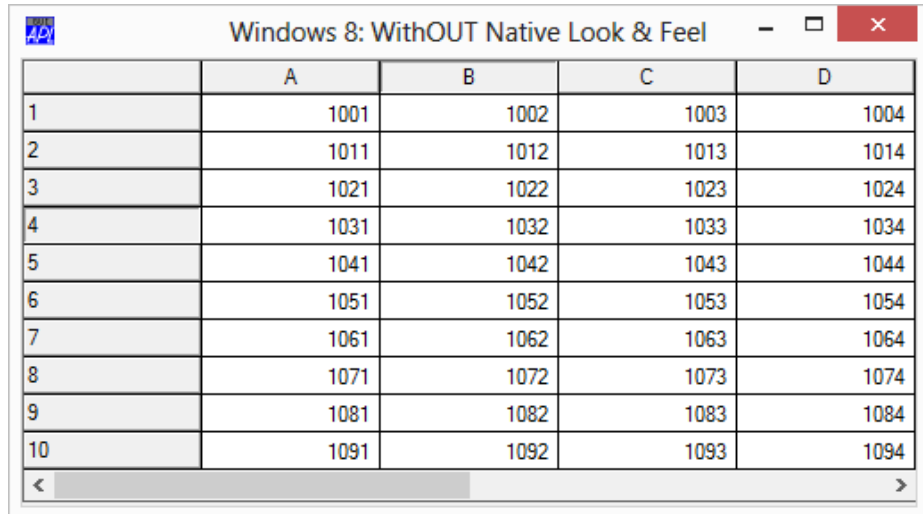
```
'F'[]WC'Form' 'Split Button'  
'F.sb'[]WC'Button' 'Select File'('Style' 'Split')
```



Change to Grid appearance (not XP)

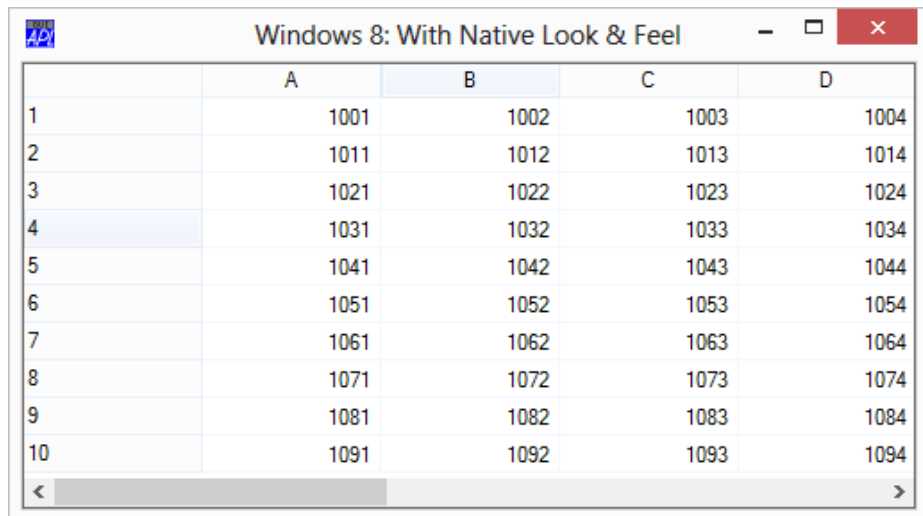
In order to make it more consistent with current Windows conventions, the default appearance of the Grid now respects the user's chosen *theme* when Native Look and Feel is enabled. In particular, the default colour of the grid lines and the shading of the row and column headings is determined by the user's theme.

This change applies to all supported Versions of Windows **except** Windows XP.



	A	B	C	D
1	1001	1002	1003	1004
2	1011	1012	1013	1014
3	1021	1022	1023	1024
4	1031	1032	1033	1034
5	1041	1042	1043	1044
6	1051	1052	1053	1054
7	1061	1062	1063	1064
8	1071	1072	1073	1074
9	1081	1082	1083	1084
10	1091	1092	1093	1094

Windows 8 default theme, Native Look and Feel NOT enabled.



	A	B	C	D
1	1001	1002	1003	1004
2	1011	1012	1013	1014
3	1021	1022	1023	1024
4	1031	1032	1033	1034
5	1041	1042	1043	1044
6	1051	1052	1053	1054
7	1061	1062	1063	1064
8	1071	1072	1073	1074
9	1081	1082	1083	1084
10	1091	1092	1093	1094

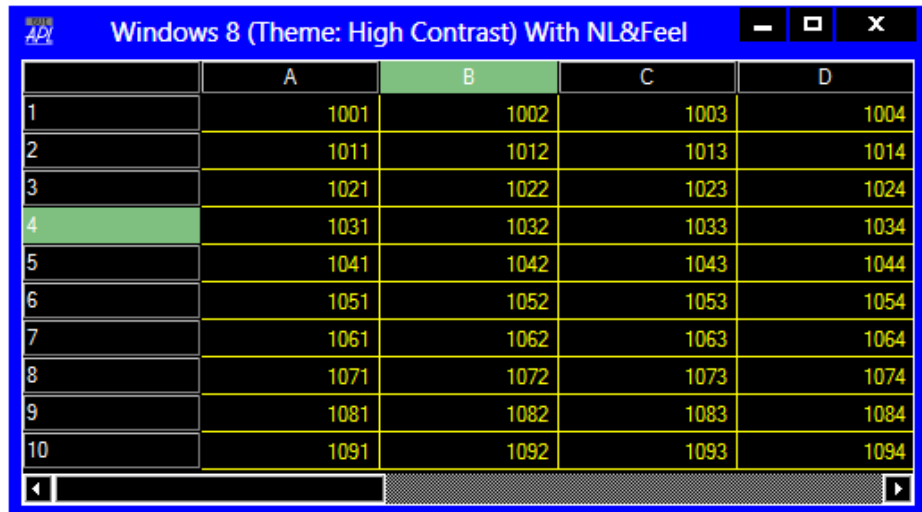
Windows 8 default Theme, Native Look and Feel enabled.

The change is more noticeable when a more colourful Theme is in use. The following pictures illustrate a Grid with and without Native Look and Feel when a Windows 8 High Contrast Theme is used.



	A	B	C	D
1	1001	1002	1003	1004
2	1011	1012	1013	1014
3	1021	1022	1023	1024
4	1031	1032	1033	1034
5	1041	1042	1043	1044
6	1051	1052	1053	1054
7	1061	1062	1063	1064
8	1071	1072	1073	1074
9	1081	1082	1083	1084
10	1091	1092	1093	1094

Windows 8 High Contrast Theme, Native Look and Feel NOT enabled.



	A	B	C	D
1	1001	1002	1003	1004
2	1011	1012	1013	1014
3	1021	1022	1023	1024
4	1031	1032	1033	1034
5	1041	1042	1043	1044
6	1051	1052	1053	1054
7	1061	1062	1063	1064
8	1071	1072	1073	1074
9	1081	1082	1083	1084
10	1091	1092	1093	1094

Windows 8 High Contrast Theme, Native Look and Feel enabled.

GridFCol, RowTitleFCol and ColTitleFCol

If Native Look and Feel is enabled, the values of the GridFCol, RowTitleFCol and ColTitleFCol properties are ignored in deference to the colours defined by the chosen theme. You may however override the theme derived colours of the grid lines using GridLineFCol.

Miscellaneous Enhancements

Dyalog APL Version 13.2 provides the following miscellaneous enhancements and changes:

`⎕FLIB` Limitation Removed

Component files that are exclusively tied were not previously reported by `⎕FLIB`. In Version 13.2 they are included in the result.

`APL_FHIST_TIE`

From Version 13.2 onwards, support for the `APL_FHIST_TIE` parameter has been removed. Component file tie time is not recorded and `⎕FHIST` always reports 0 for user ID and time in the 4th row of its result.

`⎕PROFILE`

`⎕PROFILE 'data'` and `⎕PROFILE 'tree'` have been enhanced to take an optional left argument specifying which columns of data to return.

Workspace Save

`0 ⎕SAVE ws id` no longer fails with `DOMAIN ERROR` if there are threads running or Edit/Trace windows open. These restrictions continue to apply to monadic `⎕SAVE` and `)SAVE`.

`)SAVE` and `⎕SAVE` now insist that you explicitly specify a filename. Previously if you specified just the name of a directory, the workspace was saved with the name `.DWS`.

Keyboard Shortcuts

Windows allows users to control the visibility of keyboard shortcuts.

Under Windows XP this is done from the *Display Properties* dialog box. Select the *Appearance* tab, then click the *Effects* button. Then select or clear the option labelled *Hide underlined letters for keyboard navigation until I press the Alt key*.

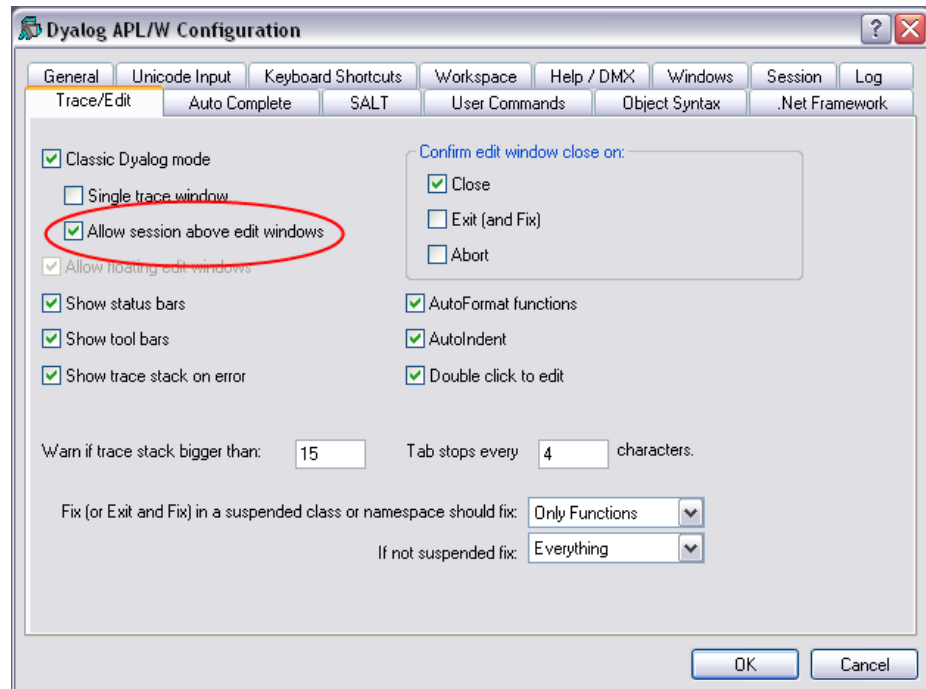
In Windows 7, open *Control Panel / Ease of Access Center / Make the Keyboard easier to use*. The option is labelled *Underline keyboard shortcuts and access keys*

The Dyalog APL GUI now honours this option. This means that the underscores in the captions of certain objects, including Label, MenuItem and Button objects, may or may not be shown until the user presses the Alt key.

Note that the actual behaviour obtained by the option is controlled by Windows and not by Dyalog.

SessionOnTop Parameter

In *Classic Dyalog* mode, it is now possible to specify whether or not the Session window may appear on top of Edit and Trace windows. To enable this setting, check the option box as shown in the picture below.



Find Objects Tool

The Find Objects dialog box now saves all your option settings between invocations *and* between APL sessions.

MakeGIF and MakePNG Methods

In Version 13.2, these methods are implemented using GDI+. This means that **MakeGIF** now generates *compressed* GIF files. In previous versions it generated *uncompressed* GIFs.

Find/Replace Tool

In previous versions of Dyalog APL the Find/Replace window was dockable. However, the resulting Window was in general unusable, with many of the fields and buttons not visible. In Version 13.2 the Find/Replace window is no longer dockable.

Announcements

Support for Version 12.1 and 13.0

Version 12.1 and Version 13.0 will continue to be supported until Version 14.0 is released, at which point both these Versions will cease to be supported.

Deprecation of 32-bit component files

Since Version 10.1, Dyalog APL has supported large span (64-bit) component files, and since Version 12.0 `⎕FCREATE` has created these by default. Existing small span (32-bit) component files are still supported and 32-bit component files may still be created if suitable options are specified, but they have restrictions which 64-bit files do not, including:

- The maximum file size is 4GB.
- The files are not fully architecture-independent meaning that there are limitations sharing them between, for example, Windows or Linux and AIX machines.
- Components may not contain Unicode data.

As announced on the Release of Version 13.0, Dyalog intends to withdraw support for 32-bit component files in the next Release which will be Version 14.0.

If you have any existing 32-bit component files, or applications which create and/or use them, Dyalog recommends that you prepare for this in the following ways:

- Ensure that Dyalog is not started with the command-line option `-F32`. This option sets the default component file type which is created to 32-bit.
- Ensure that no `⎕FCREATE` within your applications explicitly specifies that 32-bit files are to be created.
- Make plans to convert any existing 32-bit component files to 64-bit using `⎕FCOPY`. `⎕FCOPY` will create a 64-bit copy even if the file being copied is 32-bit. You may use the new user command `⎕Fto64` to locate existing small span files and convert them to the 64-bit architecture. This user command is included with Version 13.2, and can be downloaded for earlier versions as well (users of Version 13.1 can use the `⎕UUPDATE` command to download updated user commands).

Note: in order to allow the use of legacy files retrieved from backups etc., Dyalog will continue to provide read-only support for 32-bit files in Version 14.0, and in subsequent releases for a minimum of 10 years.

Auxiliary Processors

It is intended that in the next major release, Version 14.0, all Dyalog-supplied Auxiliary Processors, namely `qfsock`, `strand` and `xutils` will be removed from the product, although the interfaces for user-written Auxiliary Processors will continue to be supported. Instead Dyalog will supply equivalent DLLs/shared libraries (operating system dependent).

Random Number Generator

The Version 13.0 random number generator, that is used by Roll and Deal, is based upon the Lehmer linear congruential generator. This has several limitations, most notably that it has a limited value range of (2^{31}) . Mindful of the need to support applications that rely on the current mechanism, and the ability to generate specific repeatable random series using `▢RL`, Dyalog provided two additional random number generators in Version 13.1. Both the new algorithms support 64-bit values and both may be considered to be an improvement (in terms of randomness) over the existing mechanism. The new mechanisms were:

- Mersenne Twister random number generator. This algorithm produces 64-bit values with good distribution.
- Operating System random number generator. Under Windows APL this uses the `CryptGenRandom()` function. Under Unix/Linux it uses `/dev/urandom` [3].

You may select the random number generator in use using `16807▢`. This allows you to switch dynamically between the different algorithms if required.

In Versions 13.1 and 13.2 the default random number generator in a `CLEAR WS` is 0 (Lehmer linear congruential). **The default will be changed to 1 (Mersenne Twister) in the next release of Dyalog APL (Version 14.0)**. In preparation for this change, avoid writing code which assumes that `▢RL` will be a scalar integer.

Note: the change to the default will only impact applications if they are rebuilt from a clear workspace; saved workspaces will be unaffected.

Chapter 2:

Language Reference Changes

Assignment (Selective):

 $(EXP\ X)\leftarrow Y$

X is the *name* of a variable in the workspace, possibly modified by the indexing function $(EXP\ X[I])\leftarrow Y$, see "[Assignment \(Indexed\):](#)" on page 28. EXP is an expression that **selects** elements of X . Y is an array expression. The result of the expression Y is allocated to the elements of X selected by EXP .

The following functions may appear in the selection expression. Where appropriate these functions may be used with axis $[]$ and with the Each operator \cdot .

Functions for Selective Assignment

\uparrow	Take
\downarrow	Drop
$,$	Ravel
ϕ	Reverse, Rotate
ρ	Reshape
\succ	Disclose, Pick
Φ	Transpose (Monadic and Dyadic)
$/$	Replicate
\backslash	Expand
\square	Index
ϵ	Enlist ($\square ML \geq 1$)

Note: Mix and Split (monadic \uparrow and \downarrow), Type (monadic ϵ when $\square ML < 1$) and Membership (dyadic ϵ) may not be used in the selection expression.

Examples

```

A←'HELLO'
((Aε'AEIOU')/A)←' * '

A
H*LL*

Z←3 4ρ⌊12
(5↑,Z)←0

Z
0 0 0 0
0 6 7 8
9 10 11 12

```

```

MAT←3 3p19
(1 1QMAT)←0

MAT
0 2 3
4 0 6
7 8 0

ML←1R so ε is Enlist
names←'Andy' 'Karen' 'Liam'
(('a'=εnames)/εnames)←' * '
names
Andy K*ren Li*m

```

Each Operator

The functions listed in the table above may also be used with the Each Operator `.

Examples

```

A←'HELLO' 'WORLD'
(2↑A)←' * '
A
**LLO **RLD

A←'HELLO' 'WORLD'
((A='O')/A)←' * '
A
HELL* W*RLD

A←'HELLO' 'WORLD'
((Aε''c='LO')/A)←' * '
A
HE*** W*R*D

```

Bracket Indexing

Bracket indexing may also be applied to the expression on the left of the assignment arrow.

Examples

```

MAT←4 3p'Hello' 'World'
(2↑MAT[;1 3])←'$'
MAT
Hel$$ World Hel$$
Wor$$ Hello Wor$$
Hel$$ World Hel$$
Wor$$ Hello Wor$$

```

Assignment (Indexed):

$$\{R\} \leftarrow X[I] \leftarrow Y$$

Indexed Assignment is the Assignment function modified by the Indexing function. The phrase $[I] \leftarrow$ is treated as the function for descriptive purposes.

Y may be any array. X may be the *name* of any array or a selection from a named array ($EXP\ X$) $[I] \leftarrow Y$, see "[Assignment \(Selective\):" on page 26](#). I must be a valid index specification. The shape of Y must conform with the shape (implied) of the indexed structure defined by I . If Y is a scalar or a unit vector it will be extended to conform. A side effect of Indexed Assignment is to change the value of the indexed elements of X .

R is the value of Y . If the result is not explicitly assigned or used it is suppressed.

$\square IO$ is an implicit argument of Indexed Assignment.

Three forms of indexing are permitted.

Simple Indexed Assignment

For vector X , I is a simple integer array whose items are from the set $\iota \rho R$. Elements of X identified by index positions I are replaced by corresponding elements of Y .

Examples

```
+A←ι5
1 2 3 4 5
```

```
A[2 3]←10 ♦ A
1 10 10 4 5
```

The last-most element of Y is assigned when an index is repeated in I :

```
A[2 2]←100 101 ♦ A
1 101 10 4 5
```

For matrix X , I is composed of two simple integer arrays separated by the semicolon character (;). The arrays select indices from the rows and columns of X respectively.

Examples

```

      +B←2 3ρ'REDSUN'
RED
SUN

```

```

      B[2;2]←'O' ♦ B
RED
SON

```

For higher-order array X , \mathbf{I} is a series of simple integer arrays with adjacent arrays separated by a single semicolon character (;). Each array selects indices from an axis of X taken in row-major order.

Examples

```

      C
11 12 13
14 15 16

```

```

21 22 23
24 25 26

```

```

      C[1;1;3]←103 ♦ C
11 12 103
14 15 16

```

```

21 22 23
24 25 26

```

An indexing array may be ELIDED. That is, if an indexing array is omitted from the K th axis, the indexing vector $\iota(\rho X)[K]$ is implied:

```

      C[;1;2 3]←2 2ρ112 113 122 123 ♦ C
11 112 113
14 15 16

```

```

21 122 123
24 25 26

```

```

      C[;;]←0 ♦ C
0 0 0
0 0 0

0 0 0
0 0 0

```

Choose Indexed Assignment

The index specification **I** is a non-simple integer array. Each item identifies a single element of **X** by a set of indices with one element per axis of **X** in row-major order.

Examples

```

      C
11 12 13 14
21 22 23 24

```

```

      C[←1 1]←101 ◊ C
101 12 13 14
   21 22 23 24

```

```

      C[(1 2) (2 3)]←102 203 ◊ C
101 102 13 14
   21  22 203 24

```

```

      C[2 2ρ(1 3)(2 4)(2 1)(1 4)]←2 2ρ103 204 201 104 ◊ C
101 102 103 104
201  22 203 204

```

A scalar may be indexed by the enclosed empty vector:

```

      S
10
      S[←∅]←←'VECTOR' ◊ S
VECTOR
      S[←∅]←5 ◊ S
5

```

Choose Indexed Assignment may be used very effectively in conjunction with Index Generator (**ι**) and Structural functions in order to assign into an array:

```

      C
11 12 13 14
21 22 23 24

      ιρC
1 1  1 2  1 3  1 4
2 1  2 2  2 3  2 4

      C[1 1∅ιρC]←1 2 ◊ C
   1 12 13 14
   21  2 23 24

      C[2 ∼1∅ιρC]←99 ◊ C
   1 12 13 99
   21  2 23 99

```

Reach Indexed Assignment

The index specification **I** is a non-simple integer array, each of whose items reach down to a nested element of **X**. The items of an item of **I** are simple vectors (or scalars) forming sets of indices that index arrays at successive levels of **X** starting at the top-most level. A set of indices has one element per axis at the respective level of nesting of **X** in row-major order.

Examples

```

D←(2 3p16)(2 2p'SMITH' 'JONES' 'SAM' 'BILL')

      D
1 2 3  SMITH  JONES
4 5 6  SAM    BILL

≡J←c2 (1 2)
-3

      D[J]←c'WILLIAMS' ⋄ D
1 2 3  SMITH  WILLIAMS
4 5 6  SAM    BILL

      D[(1 (1 1))(2 (2 2) 1)]←10 'W' ⋄ D
10 2 3  SMITH  WILLIAMS
4 5 6  SAM    WILL

      E
GREEN  YELLOW  RED

      E[c2 1]←'M' ⋄ E
GREEN  MELLOW  RED

```

The context of indexing is important. In the last example, the indexing method is determined to be Reach rather than Choose since **E** is a vector, not a matrix as would be required for Choose. Observe that:

$$c2\ 1 \leftrightarrow c(c2), (c1)$$

Note that for any array **A**, **A[c⊖]** represents a scalar quantity, which is the whole of **A**, so:

```

      A←5p0
      A
0 0 0  0 0
      A[c⊖]←1
      A
1

```

Combined Indexed and Selective Assignment

Instead of X being a name, it may be a selection from a named array, and the statement is of the form $(EXP\ X)[I] \leftarrow Y$.

```

MAT←4 3p'Hello' 'World'
(2↑MAT)[1 2;]←'#'
MAT
##llo ##rld ##llo
##rld ##llo ##rld
Hello World Hello
World Hello World

MAT←4 3p'Hello' 'World'
□ML←1 A ∈ is Enlist
(∈MAT)[2×ι|0.5×ρ∈MAT]←'#'
MAT
H#l#o #o#l# H#l#o
#o#l# H#l#o #o#l#
H#l#o #o#l# H#l#o
#o#l# H#l#o #o#l#

```

Memory Manager Statistics: **$R \leftarrow \{X\} (2000\pm) Y$**

This function returns information about the state of the workspace and provides a means to reset certain statistics and to control workspace allocation. This I-Beam is provided for performance tuning and is VERY LIKELY to change in the next release.

Y is a simple integer scalar or vector containing values listed in the table below.

If X is omitted, the result R is an array with the same structure as Y , but with values in Y replaced by the following statistics. For any value in Y outside those listed below, the result is undefined.

Value	Description
0	Workspace available (a "quick" $\square WA$)
1	Workspace used
2	Number of compactions since the workspace was loaded
3	Number of garbage collections that found garbage
4	Current number of garbage pockets in the workspace
12	Sediment size
13	Current workspace allocation, i.e. the amount of memory that is actually being used
14	Workspace allocation high-water mark, i.e. the maximum amount of memory that has been used since the workspace was loaded or since this count was reset
15	Limit on minimum workspace allocation
16	Limit on maximum workspace allocation

Note that while all other operations are relatively fast, the operation to count the number of garbage pockets (4) may take a noticeable amount of time, depending upon the size and state of the workspace.

Examples

```

2000±0
55414796
2000±0 1 2 3 4 12 13 14 15 16
55414796 10121204 5 0 0 2120524 34489168 34489168 0 65536000

```

If X is specified, it must be either a simple integer scalar, or a vector of the same length as Y , and the result R is Θ . In this case, the value in Y specifies the item to be set and X its new value according to the table below.

Value	Description
2	0 resets the compaction count; no other values allowed
3	0 resets the count of garbage collections that found garbage; no other values allowed
14	0 resets the workspace allocation high-water mark; no other values allowed
15	Sets the minimum workspace allocation to the corresponding value in X ; must be between 0 and the current workspace allocation
16	Sets the maximum workspace allocation to the corresponding value in X ; 0 implies MAXWS otherwise must be between the current workspace allocation and MAXWS .

Notes:

- Note that the workspace allocation high-water mark indicates a minimum value for **MAXWS**.
- Limiting the maximum workspace allocation can be used to prevent code which grabs as much workspace as it can from skewing the peak usage result.
- Limiting the minimum workspace allocation can avoid repeatedly committing and releasing memory to the Operating System when memory usage is fluctuating.

Examples

```

      2000i2 3
6 0 33216252
      0 (2000i)2 3 14 # Reset compaction count

      2000i2 3
0 0
      30000000 40000000(2000i)15 16 # Restrict min/max ws

      (2000i)15 16
30000000 40000000
      0 (2000i)15 16 # Reset min/max ws

      (2000i)15 16
0 65536000

```

(2000I)13 14 A Current, peak WS allocation
4072532 4072532

a+10e6p'x' A Increase WS allocation

(2000I)13 14 A Current, peak WS allocation
15108580 15108580

[ex 'a' ◇ {}]wa A Decrease current WS allocation

(2000I)13 14 A Current, peak WS allocation
1962856 15108580

0 (2000I) 14 A Reset High-water mark

(2000I)13 14 A Current, peak WS allocation
1962856 1962856

Chapter 3:

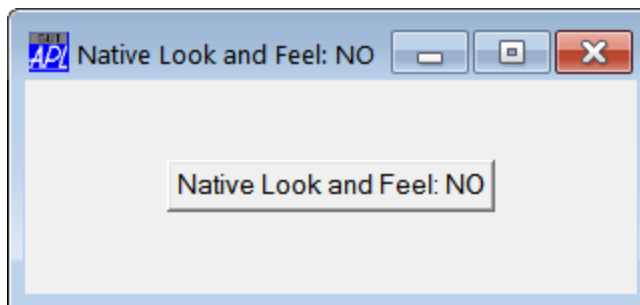
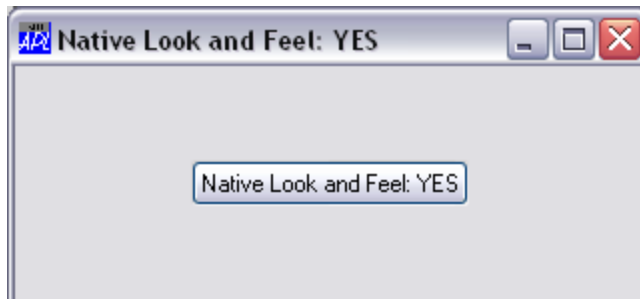
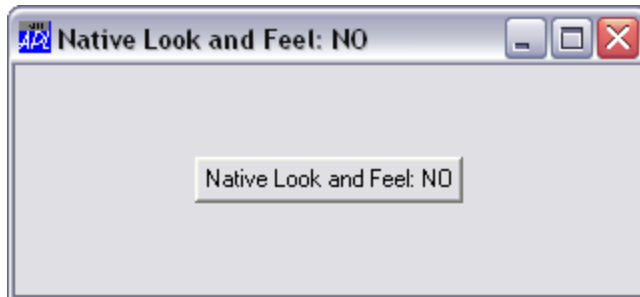
Object Reference Changes

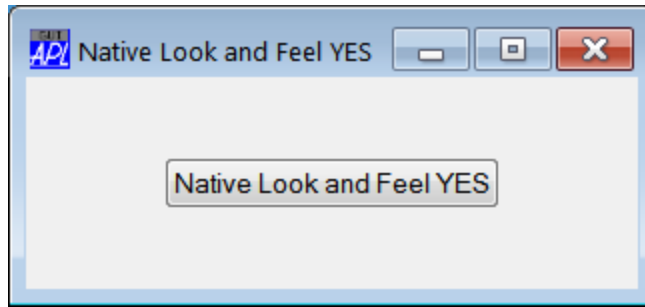
Native Look and Feel

Windows *Native Look and Feel* is an optional feature of Windows from Windows XP onwards.

If *Native Look and Feel* is enabled, user-interface controls such as Buttons take on a different appearance and certain controls (such as the ListView) provide enhanced features.

The following pictures illustrate the appearance of a simple Button created with and without *Native Look and Feel* under Windows XP and Windows 7.





Dyalog Session

During development, both the Dyalog Session and the Dyalog APL GUI will display native style buttons, combo boxes, and other GUI components if *Native Look and Feel* is enabled. The option is provided in the *General* tab of the *Configuration* dialog.

Applications

There are two ways to enable *Native Look and Feel* in end-user applications.

If you use the *File/Export...* menu item on the Session MenuBar to create a bound executable, an OLE Server (in-process or out-of-process), an ActiveX Control or a .Net Assembly, check the option box labelled *Enable Native Look and Feel* in the *create bound file* dialog box. See User Guide.

If not, set the **XPLookandFeel** parameter to 1, when you run the program. For example:

```
dyalogrt.exe XPLookAndFeel=1 myws.dws
```

Note that to have effect, *Native Look and Feel* must also be enabled at the Windows level.

ButtonEdit**Object**

Purpose:	Allows user to enter or edit data.
Parents	ActiveXControl, Form, Group, PropertyPage, SubForm
Children	Circle, Ellipse, Font, ImageList, Marker, Poly, Rect, Text, Timer
Properties	Type, Text, Posn, Size, Style, Coord, Align, Border, Justify, Active, Visible, Event, ImageListObj, Sizeable, Dragable, FontObj, FCol, BCol, CursorObj, AutoConf, Data, Attach, EdgeStyle, Handle, Hint, HintObj, Tip, TipObj, FieldType, MaxLength, Decimals, Password, ValidIfEmpty, ReadOnly, FormatString, Changed, Value, Translate, Accelerator, AcceptFiles, KeepOnClose, Transparent, ImageIndex, Redraw, TabIndex, Cue, ShowCueWhenFocused, MethodList, ChildList, EventList, PropList
Methods	Detach, ChooseFont, GetTextSize, Animate, GetFocus, ShowSIP
Events	Close, Create, FontOK, FontCancel, DragDrop, Configure, ContextMenu, DropFiles, DropObjects, Expose, Help, KeyPress, GotFocus, LostFocus, MouseDown, MouseUp, MouseMove, MouseDbClick, MouseEnter, MouseLeave, MouseWheel, Select, BadValue, KeyError, Change, DropDown

Description

The ButtonEdit object combines a single-line input field with a customisable button. It provides the same user and programmer interfaces as an Edit object (Style 'Single').

The appearance of the button, which is displayed to the right of the input field, is determined by the ImageListObj property. When clicked, the object generates a Drop-Down event. There is no default processing for this event; it is up to the programmer to take the appropriate action via a callback function.

The following picture illustrates two ButtonEdit objects

```

▽ Example;BK;White
[1] 'F'WC'Form' 'ButtonEdit'
[2] 'F.IL1'WC'ImageList'('Size' 16 16>('Masked' 1)
[3] 'F.IL1.Time'WC'Icon' 'c:\MadCap13.2\ICO\Time.ico'
[4] 'F.BE1'WC'ButtonEdit' '(30 20)(Ø 160)
[5] F.BE1.(Cue ShowCueWhenFocused)+Enter data' 1
[6] F.BE1.(ImageListObj ImageIndex)+F.IL1 1
[7]
[8] 'F.fnt'WC'Font' 'APL385 Unicode' 16
[9] BK+16 16p256lWhite+255 255 255
[10] 'F.Rotate'WC'Bitmap'('CBits'BK>('MaskCol'White)
[11] 'F.Rotate.'WC'Text' 'φ'(0 3>('FontObj'F.fnt)
[12] BK+F.Rotate.CBits
[13] 'F.IL1.'WC'BitMap'('CBits'BK>('MaskCol'White)
[14] 'F.BE2'WC'ButtonEdit' 'Hello World'(100 20)(Ø
160)
[15] F.BE2.(ImageListObj ImageIndex)+F.IL1 2
[16] F.BE2.onDropDown+Rotate'
▽

▽ Rotate msg
[1] (=msg).Text+φ(=msg).Text
▽

```



ButtonsAcceptFocus**Property**

Applies To: ToolControl

Description

This is a Boolean property that determines how the Tab key and other cursor movement keys are handled by a ToolControl object.

If ButtonsAcceptFocus is 0 (the default), when the user presses Tab or Shift+Tab to switch the input focus from another object to the ToolControl, the first ToolButton in the ToolControl receives the input focus and is highlighted. Pressing Tab or Shift+Tab again causes the input focus to move to another control. The cursor movement keys have no effect.

If ButtonsAcceptFocus is 1, when the user presses Tab or Shift+Tab to switch the input focus from another object to the ToolControl, the first or last ToolButton in the ToolControl receives the input focus and is highlighted. Note that the behaviour of Shift+Tab in this case is different. Pressing Tab or Shift+Tab again causes the input focus to move to another control, although if there is no other control to accept the input focus, it moves to the first or last ToolButton as appropriate. Pressing the cursor movement keys causes the input focus to move from one ToolButton to the next.

Cue**Property**

Applies To: ButtonEdit, Edit

Description

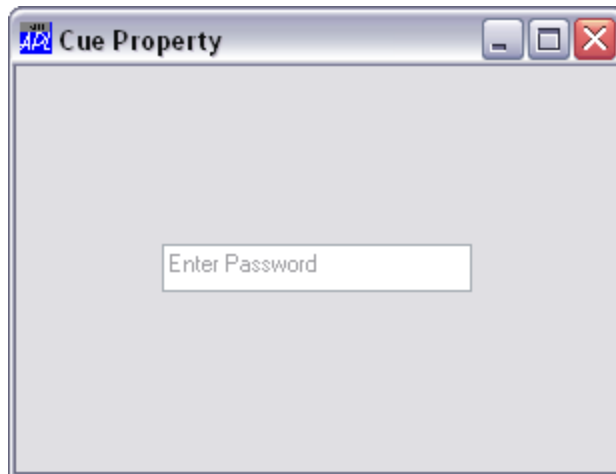
This property specifies optional text to be displayed when a ButtonEdit or an Edit object is empty. For an Edit object it applies only if the Style of the Edit object is 'Single'.

Note that this feature only applies if Native Look and Feel ([see page 38](#)) is enabled.

The Boolean property ShowCueWhenFocused determines whether or not the cue should also be displayed once the user has tabbed into or clicked on the input field (and thus given it the focus).

Example

```
'F' □WC 'Form' 'Cue Property'  
'F.E' □WC 'Edit'  
F.E.Cue←'Enter Password'
```



HeaderImageIndex

Property

Applies To: ListView

Description

The HeaderImageIndex property is an integer vector that specifies the images to be displayed alongside each column heading in a ListView object in Report View. Each positive element of HeaderImageIndex specifies an index into the ImageList object specified by the HeaderImageList property. The special values `-1` and `-2` specify the standard *Sort Up* and *Sort Down* images respectively.

HeaderImageList

Property

Applies To: ListView

Description

The HeaderImageList property specifies the name of or ref to an ImageList object that contains images to be displayed alongside each column heading in a ListView object in Report View.

HighlightHeaders**Property**

Applies To: Grid

Description

The HighlightHeaders property is a Boolean value (default 1) that specifies whether or not the appropriate row and column titles in a Grid are highlighted corresponding to the currently selected block of cells.

HThumbDrag**Event 442**

Applies To: Form, SubForm

Description

If enabled, this event is generated when the user attempts to drag the thumb in a horizontal scrollbar in a Form or SubForm. This event occurs only in a Form or SubForm whose HScroll property is set to `-1` and is distinct from the Scroll event that is generated by a Scroll object.

The event message reported as the result of `□□DQ`, or supplied as the right argument to your callback function, is a 3-element vector as follows:

[1]	Object	ref or character vector
[2]	Event	'HThumbDrag' or 442
[3]	Position	numeric

The value of Position is the new (requested) position of the Thumb. Setting the action code of this event to `-1`, or returning a 0 from a callback function attached to it, has no effect.

Masked**Property**

Applies To: ImageList

Description

The Masked property specifies whether or not the ImageList will contain opaque or transparent images. It may be 0, 1 (the default) or 2.

Masked must be established when the ImageList is created by `ImageList` and may not subsequently be altered. An inappropriate value of Masked will cause the images to be drawn incorrectly.

If Masked is 0, the ImageList expects opaque Bitmap objects.

If Masked is 1, the ImageList expects low-colour (4-bit or 8-bit) Icon objects whose transparency is defined by their Mask property.

If Masked is 2, the ImageList expects Bitmap or Icon objects whose alpha channel (the degree of transparency of each pixel) is encoded in their CBits property, along with the colours.

PageSize**Property**

Applies To: Form, Scroll, SubForm

Description

For a Form and SubForm, the PageSize property is a 2-element integer vector which specifies the size of the thumb in the vertical and horizontal scrollbars respectively.

For a Scroll object it is a single integer.

If PageSize is 0 (the default) it specifies the default thumb. Otherwise, PageSize is expressed in proportion to the corresponding value of Range. For example, if Range is 1000, setting PageSize to 100 will obtain a thumb which is approximately 10% of the height or length of the scrollbar.

ReportBCol**Property**

Applies To: ListView

Description

In Report View, the ReportBCol property is either a scalar or a matrix that specifies the background colours for each item displayed in a ListView object .

Its first column refers to the Items themselves, and subsequent columns to the elements of ReportInfo.

i.e. if non-scalar, $(\rho\text{ReportBCol}) \leftrightarrow (0 \ 1 + \rho\text{ReportInfo})$

Each element of ReportBCol is either an integer colour value or a 3-element of RGB colour indices.

ReportImageIndex**Property**

Applies To: ListView

Description

The ReportImageIndex property is an integer scalar or matrix that specifies the images to be displayed alongside each item in a ListView object in Report View.

If it is a matrix, its first column specifies the indices of the icons to be displayed against the Items of the ListView, overriding the icons specified by ImageIndex, and its subsequent columns specify the indices of the icons to be displayed against the elements of ReportInfo.

i.e. if non-scalar, $(\rho\text{ReportImageIndex}) \leftrightarrow (0 \ 1 + \rho\text{ReportInfo})$

Each element of ReportImageIndex specifies an index into the ImageList object specified by the ImageListObj property.

RowHiddenDepth**Property**

Applies To: Grid

Description

The RowHiddenDepth property identifies which rows of a Grid are currently hidden.

SelectionBorderWidth**Property**

Applies To: Grid

Description

The SelectionBorderWidth property specifies the width of the border that is drawn around the currently selected block of cells. It is expressed in pixels.

SelectionColor**Property**

Applies To: Grid

Description

The SelectionColor property specifies the colour used to highlight the currently selected block of cells and, if HighlightHeaders is 1, the corresponding row and column headings. See also ["SelectionColorAlpha" on page 48](#).

SelectionColorAlpha**Property**

Applies To: Grid

Description

The SelectionColorAlpha property is a 2-element integer vector that specifies the degree of transparency or shade of the colour that is used to highlight the currently selected block of cells in a Grid. See ["SelectionColor" on page 48](#).

The first element refers to the shade to be used when the Grid has the input focus; the second to when it doesn't. Each element is an integer in the range 0 (invisibly light) to 255 (fully dark).

ShowBalloonTip**Method 860**

Applies To: SysTrayItem

Description

The ShowBalloonTip method displays a BalloonTip in a SysTrayItem object.

The argument to ShowBalloonTip is a 1, 2, 3 or 4-element array as follows:

[1]	Title	character vector
[2]	Text	character vector or matrix
[3]	Icon	Integer scalar, a character vector or a ref
[4]	Flags	Integer

The *Title* parameter is the text to be displayed in the BalloonTip title (maximum length 64).

The *Text* parameter is the text (maximum length 256) to be displayed in the BalloonTip. If omitted or empty, the BalloonTip is not displayed.

If the *Icon* parameter is an integer, it means:

0	No icon
1	Information icon
2	Warning icon
3	Error icon

Other values represent the name or a ref to an Icon object. If the *Icon* parameter is omitted, no icon is displayed in the BalloonTip.

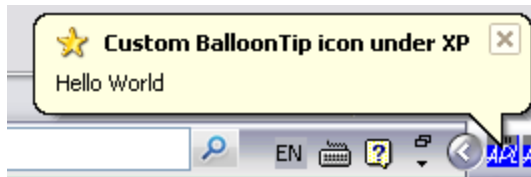
If the *Icon* parameter specifies a *large* Icon object (32 x 32 bits) the Flags parameter must be 32. Otherwise this parameter is not used.

Windows XP

Under Windows XP, only small (16x16) icons are supported. Furthermore, the BalloonTip uses the icon associated with the SysTrayItem itself regardless of the custom icon specified by ShowBalloonTip. However, the following code provides a work-around, which is to switch the icon for the SysTrayItem itself to the desired icon temporarily, just for the invocation of the ShowBalloonTip method.

Example (XP)

```
▽ XPBalloonTip;tmpIcon;text;title
[1] 's'\WC'SysTrayItem' A default (APL) icon
[2] 'star'\WC'Icon'('Shell32.dll' -43)
[3] tmpIcon←s.IconObj
[4] s.IconObj←star
[5] text←'Hello World'
[6] title←'Custom BalloonTip icon under XP'
[7] s.ShowBalloonTip title text star
[8] s.IconObj←tmpIcon
▽
```

**ShowCueWhenFocused****Property**

Applies To: ButtonEdit, Edit

Description

This Boolean property specifies whether or not the text specified by the property should be displayed once the user has tabbed into or clicked on the empty input field (and thus given it the focus). For an Edit object it applies only if the Style of the Edit object is 'Single'.

VThumbDrag**Event 441**

Applies To: Form, SubForm

Description

If enabled, this event is generated when the user attempts to drag the thumb in a vertical scrollbar in a Form or SubForm. This event occurs only in a Form or SubForm whose HScroll property is set to `-1` and is distinct from the Scroll event that is generated by a Scroll object.

The event message reported as the result of `□□DQ`, or supplied as the right argument to your callback function, is a 3-element vector as follows:

[1]	Object	ref or character vector
[2]	Event	'VThumbDrag' or 441
[3]	Position	numeric

The value of Position is the new (requested) position of the Thumb. Setting the action code of this event to `-1`, or returning a 0 from a callback function attached to it, has no effect.

Index

A

Account 9
aedit User Command 5
Array Editor 3, 5
array separator 28
assignment
 indexed 28
 selective 26

B

ButtonEdit 40
ButtonsAcceptFocus 42

C

choose indexed assignment 30
Cue 43

E

Events
 HThumbDrag 45
 VThumbDrag 51

F

find and replace dialogs 21
Find Objects Tool 20
Fto64 User Command 22

H

HeaderImageIndex 44
HeaderImageList 44
HighlightHeaders 45
HThumbDrag 45

I

i-beam
 memory manager statistics 33
indexed assignment 28

K

Key Features 1

M

MakeGIF method 21
MakePNG method 21
Masked 46
MAXWS parameter 10, 34
memory manager statistics 33
Methods
 ShowBalloonTip 49
Miscellaneous Enhancements 19

N

Native Look and Feel 38

O

Objects
 ButtonEdit 40

P

PageSize 46
Properties
 ButtonsAcceptFocus 42
 Cue 43
 HeaderImageIndex 44
 HeaderImageList 44
 HighlightHeaders 45
 Masked 46
 PageSize 46
 ReportBCol 47
 ReportImageIndex 47
 RowHiddenDepth 47
 SelectionBorderWidth 48

SelectionColor 48
SelectionColorAlpha 48
ShowCueWhenFocused 50

R

reach indexed assignment 31
ReportBCol 47
ReportImageIndex 47
RowHiddenDepth 47

S

SelectionBorderWidth 48
SelectionColor 48
SelectionColorAlpha 48
selective assignment 26
SessionOnTop parameter 20
ShowBalloonTip 49
ShowCueWhenFocused 50
simple indexed assignment 28
System Requirements 4

U

User Commands
 aedit 5
 Fto64 22
 UUPDATE 22
UUPATE User Command 22

V

VThumbDrag 51

X

XP Look and Feel 38