

## **J User**



**Chris Burke**

Copyright © 1991-2002 Jsoftware Inc. All Rights Reserved.

Last updated: 2002-10-4

[www.jsoftware.com](http://www.jsoftware.com)

# Table of Contents

1	<a href="#"><u>J 5.01 Release Highlights and Overview</u></a>
2	<a href="#"><u>Old Windows Release Notes</u></a>
3	<a href="#"><u>J User License</u></a>
4	<a href="#"><u>General Information</u></a>
5	<a href="#"><u>About J</u></a>
6	<a href="#"><u>Products</u></a>
7	<a href="#"><u>J User License Order Form</u></a>
8	<a href="#"><u>Support and Questions</u></a>
9	<a href="#"><u>Copyright / Warranty / License</u></a>
10	<a href="#"><u>Starting J</u></a>
11	<a href="#"><u>Mac J.402 Startup</u></a>
12	<a href="#"><u>Directory Paths</u></a>
13	<a href="#"><u>Session Manager</u></a>
14	<a href="#"><u>Session Manager</u></a>
15	<a href="#"><u>Execution Windows</u></a>
16	<a href="#"><u>Script Windows</u></a>
17	<a href="#"><u>Input Log</u></a>
18	<a href="#"><u>Menus</u></a>
19	<a href="#"><u>Status Bar</u></a>
20	<a href="#"><u>Controlling</u></a>
21	<a href="#"><u>Script Libraries</u></a>
22	<a href="#"><u>system\main</u></a>
23	<a href="#"><u>system\packages</u></a>
24	<a href="#"><u>system\classes</u></a>
25	<a href="#"><u>system\extras</u></a>
26	<a href="#"><u>system\examples</u></a>
27	<a href="#"><u>user</u></a>
28	<a href="#"><u>temp</u></a>
29	<a href="#"><u>Directory Layout</u></a>
30	<a href="#"><u>Script Library Overview</u></a>
31	<a href="#"><u>scriptdoc utility</u></a>

32 [bmp](#)  
33 [colib](#)  
34 [color16](#)  
35 [colortab](#)  
36 [compare](#)  
37 [convert](#)  
38 [coutil](#)  
39 [csv](#)  
40 [dates](#)  
41 [dd](#)  
42 [debug](#)  
43 [dir](#)  
44 [dll](#)  
45 [files](#)  
46 [format](#)  
47 [graph](#)  
48 [isigraph](#)  
49 [jfiles](#)  
50 [jmf](#)  
51 [jselect](#)  
52 [keyfiles](#)  
53 [kfiles](#)  
54 [menu](#)  
55 [misc](#)  
56 [myutil](#)  
57 [nfiles](#)  
58 [numeric](#)  
59 [pack](#)  
60 [parts](#)  
61 [plot](#)  
62 [print](#)  
63 [publish](#)  
64 [random](#)  
65 [regex](#)

66 [rgb](#)  
67 [socket](#)  
68 [statdist](#)  
69 [statfns](#)  
70 [stdlib](#)  
71 [strings](#)  
72 [sysenv](#)  
73 [text](#)  
74 [trig](#)  
75 [validate](#)  
76 [viewmat](#)  
77 [winapi](#)  
78 [winlib](#)  
79 [write](#)  
80 [Definition Summaries](#)  
81 [Definitions by Script](#)  
82 [Development Environment](#)  
83 [Menu Commands](#)  
84 [Component Files](#)  
85 [Keyed Files](#)  
86 [Find in Files](#)  
87 [Printing](#)  
88 [Debug](#)  
89 [Overview](#)  
90 [Verbs](#)  
91 [Commands](#)  
92 [Stops](#)  
93 [Watch](#)  
94 [Project Manager](#)  
95 [Overview](#)  
96 [Project File](#)  
97 [Project Manager](#)  
98 [Project Manager Tabs](#)  
99 [Building Applications](#)

100 [Project Conventions](#)  
101 [Locked Scripts](#)  
102 [Window Driver](#)  
103 [Overview](#)  
104 [wd](#)  
105 [Window Forms](#)  
106 [Event Handlers](#)  
107 [wdhandler](#)  
108 [Entering Information](#)  
109 [Form Locales](#)  
110 [Other Message Handlers](#)  
111 [Wait](#)  
112 [System Events](#)  
113 [Window Controls](#)  
114 [Overview](#)  
115 [Parent Windows](#)  
116 [Location and Size](#)  
117 [Child Controls](#)  
118 [Child Classes](#)  
119 [Richedit Control](#)  
120 [Statusbar](#)  
121 [Tab Control](#)  
122 [Toolbar](#)  
123 [Common Dialog Boxes](#)  
124 [Fonts](#)  
125 [Accelerator Keys](#)  
126 [Menus](#)  
127 [Tab and Cursor Keys](#)  
128 [Ownerdraw](#)  
129 [Window Driver Command Reference](#)  
130 [wd commands](#)  
131 [gl2 commands](#)  
132 [fontspec](#)  
133 [isigraph events](#)

134 [Mapping Mode](#)  
135 [gl3 commands](#)  
136 [OpenGL printing](#)  
137 [Form Editor](#)  
138 [Overview](#)  
139 [Hints](#)  
140 [Mouse](#)  
141 [Keyboard](#)  
142 [Design](#)  
143 [New Control](#)  
144 [Control](#)  
145 [Parent](#)  
146 [Menu](#)  
147 [Toolbar](#)  
148 [Statusbar](#)  
149 [Code](#)  
150 [Tab Order](#)  
151 [Run](#)  
152 [Defaults](#)  
153 [Tech Notes](#)  
154 [Regular Expressions](#)  
155 [Regular Expression](#)  
156 [Patterns](#)  
157 [Verbs](#)  
158 [Utilities](#)  
159 [Demo](#)  
160 [Copyright](#)  
161 [Grid Control](#)  
162 [Overview](#)  
163 [Classes](#)  
164 [Methods](#)  
165 [Properties](#)  
166 [Actions](#)  
167 [Plot](#)

168 [Overview](#)  
169 [pd verb](#)  
170 [plot verb](#)  
171 [Plot Class](#)  
172 [Plot Types](#)  
173 [Plot Commands](#)  
174 [Plot Options](#)  
175 [Plot Data](#)  
176 [Plot Colors](#)  
177 [OpenGL](#)  
178 [Movement Keys](#)  
179 [Viewing](#)  
180 [Java](#)  
181 [Java](#)  
182 [Java jserver class](#)  
183 [Java classpath](#)  
184 [Jsoftware Java applets](#)  
185 [Java examples](#)  
186 [Java applet security](#)  
187 [Sockets](#)  
188 [Socket Driver](#)  
189 [Socket Utilities](#)  
190 [J Socket Protocol](#)  
191 [DDE](#)  
192 [DDE Overview](#)  
193 [Server and Client](#)  
194 [DDE Conversations](#)  
195 [J Commands & Events](#)  
196 [Communication Protocol](#)  
197 [Examples](#)  
198 [DLLs and Memory Management](#)  
199 [Calling DLLs](#)  
200 [cd Domain Error & GetLastError](#)  
201 [Memory Management](#)

202 [Calling J.DLL](#)  
203 [ODBC Data Driver](#)  
204 [Overview](#)  
205 [The SQL Language](#)  
206 [Installing ODBC](#)  
207 [Connection & Statement Handles](#)  
208 [Data Driver](#)  
209 [Listing the Data Sources](#)  
210 [ODBC error messages](#)  
211 [Data Source Connection](#)  
212 [Selecting & reading data](#)  
213 [Updating a record](#)  
214 [Creating a new file](#)  
215 [SQL Statements](#)  
216 [SQL Elements](#)  
217 [SQL Reserved Words](#)  
218 [OLE & OCX](#)  
219 [Overview](#)  
220 [J OLE Automation Server](#)  
221 [J OLE/OCX Client](#)  
222 [Examples](#)  
223 [Tutorial: J OLE Server for Excel](#)  
224 [Tutorial: J OLE Client to Excel](#)  
225 [Labs](#)  
226 [Overview](#)  
227 [Lab Header](#)  
228 [Lab Sections](#)  
229 [Running Labs](#)  
230 [Lab Author](#)  
231 [Rich-Text](#)  
232 [Program Access](#)  
233 [Index](#)



## **J 5.01 Release Highlights and Overview**

For installation info see Windows or Unix.

J has a console interface called Jconsole and a GUI interface called Jwd. Use Jwd for an easier, user-friendly introduction to both standard and new features. In Windows, start Jwd with the blue J icon created in program group J by the install. In Unix, start Jwd by running jw in the J directory (for example, ~/j501a/jw).

The system includes 3 online books that are introductions: Learning J by Roger Stokes, and [J User License \(JUL\) p3](#) to program in J.

J can be installed on any number of platforms and systems for free. All J installations are complete and there are no 'crippled' versions. Debug, performance, and other features that previously required a software license fee for each installation are now available on all J installations.

End user applications (runtime apps) built with J that don't provide a J programming interface can be used without a JUL.

### **Incompatible changes**

The 501 release is a major release (4 to 5). In minor releases (4.05 to 4.06) we do our best to minimize changes which will break existing applications. In a major release we are more cavalier and introduce incompatible changes if we feel there is sufficient long term benefit.

We have concluded that the tridents (except for fork) and some of the bidents complicate the system out of proportion to their utility and they are decommitted. See "J 5.01 Non-compatible Changes" in red J icon.

In Unix, start Jwd by running jw in the J directory (e.g., ~/j501a/jw). This can be made easier by creating an icon, copying jw to a directory in the path, creating a link, etc.

### **Jconsole**

Jconsole is a command line interface to the JE. It does not implement wd. Jconsole is very similar in Windows and Unix and can be run from a host console and can be used like any other console command. The jconsole binary in Windows is called jconsole.exe and in Unix it is called jconsole.

In Windows, start Jconsole by running jconsole.exe. For example, if J is installed in directory c:\j501a: click Start taskbar button, select Run, and type c:\j501a\jconsole. Or you can type that line in any DOS console window. If you use Jconsole a lot you may want to create a bat file (perhaps called jc.bat) in a directory in PATH.

In Unix, start Jconsole by running jc in the J directory (e.g., ~/j501a/jc). This can be made easier by creating an icon, copying jc to a directory in PATH, creating a link, etc. Hash bang (!) scripts can use Jconsole. Jconsole uses readline and identifies itself as 'jconsole'. See the Unix man or info pages on readline.

## **Script Library**

J definitions are stored in ASCII plain text files called scripts. J scripts usually have a filename suffix of ijs. The system includes an extensive script library and related files that include an IDE, tutorials, demos, tools, and utilities.

## **profile**

When a JFE starts it loads profile.ijs from the library to initialize the system. The Jwd profile uses wd to create a complete IDE.

## **Interactive Development Environment**

The same Jwd IDE is available in Windows and Unix. The IDE is written in J/wd and is implemented in library scripts. It is open and available for study and modification. The IDE in previous versions was closed and was implemented in C++.

In previous Windows versions the IDE was based on MDI (multi-document interface) and the edit and execution windows existed within a single application frame. The MDI model is dropped in this release and edit and execution windows are independent top-level windows. Some users who have grown up with MDI will

miss it, but we believe there are benefits to the new approach. It is simpler, does away with the requirement for MDI support, and is more inline with current IDE's in other languages.

## **Context sensitive help**

Ctrl+F1 in an ijs or ijs window displays the Dictionary page for the primitive at the caret.

Ctrl+F1 with the caret in the last line of a script error report opens the script and highlights the error. An example of a script error report is:

```
load 'c:\j501a\user\test.ijs'  
|domain error: script  
|  'a'      +2  
|[-5] c:\j501a\temp\2.ijs
```

## **Portability**

J is portable across platforms. An application written for Jconsole or Jwd is as close as you can get to 'write once, run anywhere'. The JE is as identical as possible and wd, except for a few Windows specific features, is portable.

A Jwd application developed in Unix will run in Windows on either Jwdw or Jwdp.

A Jwd application developed in Windows in either Jwdp or Jwdw will run in Unix as long as the Windows specific features of Jwdw are not used.

For information on Jwdp in Windows see red J icon.

## **J Engine Protocol**

The J Engine Protocol allows any client with sockets to have full use of a JE. This provides facilities that are similar to J OLE Automation in Windows, but does so in a portable, open, more efficient, and much simpler manner. See [J Engine Protocol p190](#) for documentation.

## **Index**

The index for the online help books has been stripped down to a minimum. Previously it indexed everything and was so large that it was slow enough to be a nuisance. Use Edit|Find In Files for a general and powerful search of html help.

## **PDF Books for Hardcopy**

www.jsoftware.com has downloadable PDF versions of all the J books that are included in online html format with the release. The PDF format is particularly useful for printing hardcopy pages, chapters, or even entire books. There are additional publications available at the web site.

## **ctrl+Break, Break, and ctrl+C**

You can interrupt J execution. In Jwdw you signal with ctrl+Break (key labeled as Pause/Break); in Jwdp you signal with Break, and in Jconsole you signal with ctrl+C.

One signal interrupts at the start of a sentence with an attention interrupt or a request for input (for example, 1!:[1) with an input interrupt. Input interrupt is not currently supported in Jconsole in Unix.

Two or more signals breaks execution in the middle of a sentence with a break

It is much preferred to use a single signal to get an attention interrupt as the state of execution at the start of a line is clear. In a break it is not clear what has been executed and there is some chance of crash as it is difficult to ensure a clean state in all cases. Signals should be made carefully and you should wait many seconds before making additional signals.

## **wd changes**

There are numerous wd changes. See [Window Driver Command Reference Overview p129](#) for documentation.

## **gl2 changes - wd 2d graphics**

load'gl2' now loads into the jgl2 locale. This keeps the many gl2 definitions from cluttering up the z locale. Production users of gl2 can either use the full name (e.g. glline\_jgl2\_) or can add the jgl2 locale to their locale path. Requiring \_jgl2\_ on all names for casual use is a nuisance and you can use coinsert to add jgl2 to your current path. For example: coinsert'jgl2' will allow you to use gllines in base without having to add the \_jgl2\_.

gl2 mapping has been changed and simplified. There are 3 standard mapping modes:

glmap MM\_DEFAULT - x right, y up, scales glwindowext to fit

glmap MM\_RAW - x right, y down, units are pixels

glmap MM\_RIGHTDOWN - x right, y down, scales glwindowext to fit

New gl command 11!:2999 takes a list of multiple gl commands. Each command starts with an integer count followed by the command and data. For example:

11!:2999 [ 4 2013 500 500 4 2013 900 100 2 2036 NB. glline, glline, glshow

11!:2999 [ 4 2056 500 200 5 2038 65 66 67 2 2036 NB. gltextxy, gltext, glshow

This can be used to move the overhead in the J Engine Protocol of passing thousands of individual small commands over the socket interface by one large 2999 command.

Decommitted: glmapraw.

New commands for working with pixels:

glpixels x y w h pixeldata

pixeldata is an integer per pixel with RGB values

pixeldata =: glqpixels x y w h

See [gl2 Command Reference p131](#) for documentation.

## Foreigns

Decommitted:

1!:40 1!:41 1!:42

2!:4 2!:6

9!:30 9!:31

New:

1!:43 returns current working directory (getcwd posix)

1!:44 sets current directory (chdir posix)

1!:45 returns default profile (for example: c:\j501a\profile.ijjs)

## Java version

Jwdp (Java based Jwd) has been developed and tested with Java version 1.4.0.

## Linux

fvwm2 is NOT supported. You must run Gnome, KDE, or other X window managers.

## jservlet and jtelnet classes

The jservlet class allows you to create and manage a separate J task as a server. jservlet.ijjs (open 'jservlet') defines a jservlet class that uses the J Engine Protocol to create and manage a J Engine server. This class works in both Jconsole and Jwd. It starts a new J task, either on the local machine or a remote machine, and controls that task with the J Engine Protocol. You can take advantage of multiple processors on the same host, remote hosts, and build applications where the interactive GUI runs in one JE and the data processing is done in others as appropriate. For example:

```
require'jservlet'
js=: conew'jservlet'    NB. create new jservlet object
local__js"             NB. create J server task
run__js'abc=: i.2 3 4'  NB. run sentence in server task
d=: get__js'abc'        NB. get value from J server task
destroy__js"           NB. destroy object and free resources
```

The jservlet class also supports J server tasks running on other machines. It uses jtelnet to start the J server on the remote machine. If you had J installed on another machine that was accessible to you from telnet you could start and use a J server on that remote machine. For example, if your other machine is called Frodo and you have a user id of Bilbo and a password of Baggins, you could try the following:

```

require'jsserver'
js=.conew'jsserver'
NB. start J server on remote machine
remote__js'Frodo Bilbo Baggins 0'
NB. run sentence on remote server
run__js'abc=:i.2 3 4'
destroy__js' '

```

You could use Frodo's IP address (e.g. 192.168.1.5) instead of the name.

The remote jsserver uses jtelnet, a simple and naive implementation of the telnet protocol. For example:

```

require'jtelnet'
tn=: conew'jtelnet'      NB. create jtelnet object
logon__tn'Frodo Bilbo Baggins 0'
run__tn'ls'
destroy__tn' '

```

Open and study the scripts for more information and ideas on their use.

## Command line parameters

With -jprofile, -jijx, and -js command line parameters you can start J in various ways. When J starts the JFE puts all the command line parameters into noun ARGV\_z\_.

In the following J is j.exe or jconsole.exe or jw or jc. FN is the name of a script. ARGS is 0 or more additional command line parameters.

J - JFE loads profile

J FN - JFE loads profile; profile loads FN

J FN ARGS - JFE loads profile; profile loads FN; ARGS available in ARGV

J -jprofile - no profile (default ijx window if Jwd)

J -jprofile FN - JFE loads FN (alternate profile)

J -jprofile FN ARGS - JFE loads FN

J -jijx FN - JFE loads profile; profile does not create ijx and loads FN

J -jijx FN ARGS - JFE loads profile; profile does not create ijx and loads FN

J -js ARGS - creates verb ARGVERB\_z\_ from ARGS and runs it

Examples with -js:

```
jconsole.exe -js a=.23 b=.3 "echo a*b"
```

```
jconsole.exe -js a=.23 b=.3 "echo a*b" exit
```

An end user application (runtime app) can be started with -jijx if it needs the standard profile, or with -jprofile FN if it doesn't require profile.

Jwd tests at the end of the execution of every sentence. If there is an error and no ijx window, then a message is displayed and the session is terminated after the message is closed. If there are no forms, then the session is terminated.

## Unix J #! script

A #! J script (hash bang J script) is an executable text file with a first line that gives the full path the jconsole binary. Try the following:

create file sumsquares with 3 lines of text:

```
#!/usr/local/bin/jconsole  
echo +/*:0".>,.2}.ARGV  
exit''
```

make it executable ( `chmod +x` ) and run it

```
./sumsquares 1 2 3 4 5
```

Use `NB.` to comment out the `exit''` to stay in J.

The following loads profile, which loads the #! script, which echos the result on the console and leaves J running.

```
#!/bin/jconsole  
load'strings'  
echo 'abcXXXdef' rplc 'XXX'; insert '
```

The following is the same, except it exits J at the end.

```
#!/bin/jconsole
```



```
load'strings'
echo 'abcXXXdef' rplc 'XXX';' insert '
exit"
```

The following doesn't load profile and just loads the script.

```
#!/jconsole -jprofile
...
```

Profile loads `jconsole` with the following definitions that are useful in `#!/J` scripts:

```
ARGV - boxed list of jconsole, script name, and arguments
echo - format and display output
getenv - get value of environment variable
stdin - read from standard input
stdout - write to standard output
stderr - write to standard error
exit - exit J (arg is return code)
```

`stdin` is defined with `stdout` as its obverse (see the `:.` conjunction). When used with `&.` (under conjunction), as in `foo&.stdin ''` `stdin` is first called, reading all of standard input. That input is the argument to `foo`, and the result is passed to the inverse of `stdin`, which is `stdout`. A verb which transforms a character list can be combined with the `stdin` verb with `under` to apply the transformation as a Unix filter. As an example we will create a Unix filter which reverses all the characters in a file. Rather than just using `|.` we'll use `(|.@): , {:` which reverses all but the last character, and appends the last character to it. For files which end in a newline, this reverses the file keeping that newline at the end. Define the `#!/J` script `reverse` as follows:

```
#!/usr/local/bin/jconsole
rev=. |.@}: , {:
rev&.stdin ''
exit''
```

If you wanted to do a complete reverse of a file which does not end in a newline you could do the following:

```
rev=. |.`(|.@}: , {: )@.(LF&=@{: )
```

`echo` uses `1!:2` to write to J output (file number 2) and formats and writes any J array. `stdout` and `stderr`, however, must be given character lists, and writes

them unaltered. In particular, `echo 'a line'` will write a trailing newline character whereas `stdout 'a line'` does not.

## Unix - jconsole - stdin and stdout

The verb defined below calls a program, writes to its standard input, and reads its output.

```
run=: 4 : 0
'p o i'=. 2!:2 x.    NB. Run command, save Process, Output, Input
y. fwrite i          NB. Write to its input
fclose i             NB. Close its input
2!:3 p               NB. Wait for process to terminate
z=.fread o           NB. Read its output
fclose o             NB. Close its output
z                    NB. Result
)
```

## Starting J - tech details

In Windows, 1!:45 returns profile.ijs in the path of the J Front End (j.exe or jconsole.exe). For example: c:\j501a\profile.ijs.

In Unix, 1!:45 uses environment variables and the J version to determine the default profile. The version is the text in 9!:14" up to the first /. If 9!:14" returned j501a/2002-07-05/17:50, then the version is j501a. If HOME/version/profile.ijs exists, then it is the default profile. For example, if HOME was /home/eric, then /home/eric/j501a/profile.ijs would be the default profile if it existed. If that file doesn't exist, environment variable JPATHversion (for example, JPATHj501a), if it is defined, is the default profile.

Normally J is initialized by the JFE with:

```
(3 : '0!:0 y.')<1!:45''[ARGV_z_=:...
```

The JFE command line is given to the JE by setting ARGV\_z\_. The 1!:45" returns the full path to the default profile. The profile is loaded by an explicit verb and it must use =: for global assignments.

The default profile defines PROFILE\_z\_ (if not already defined) as 1!:45". This makes it easy for a stub profile to redirect to another profile.

-jprofile as the first parameter with additional parameters initializes J with:

```
( 3 : ' 0 ! : 0 y . ' ) 2 { ARGV _ z _ = : . . .
```

That is, the parameter after -jnoprofile is loaded instead of the standard profile.

## Previous Release Highlights

See [Old Windows Release Notes p2](#) for documentation from previous releases that has not yet been merged into the standard documentation, or perhaps just stands out more clearly.

## Windows install

J is installed on your windows system with program j501X.exe (where X indicates a bug release level). You can download this program from [www.jsoftware.com](http://www.jsoftware.com). Run this program and follow the instructions to install J. Start J with the blue J icon the install creates in program group J.

## Windows Jwdp red J icon

You should normally run Jwdw (blue J icon) as it has the following advantages over Jwdp:

1. supports OCX and OLE automation (not available in Jwdp)
2. starts quickly (Java is slow to start)
3. don't need to install/maintain Java Runtime Environment
4. is more stable (Jwdp is new and more likely to have bugs)
5. Java is a bit flakey compared to Win32
6. will run applications developed in Jwdp
7. develop apps for Jwdp by simply avoiding Windows only features

You may want to run Jwdp to see more closely how your application will look in Unix, or to check more carefully that it really is portable, or just because you are curious.

To run Jwdp you need to have the Java Runtime Environment or the Java SDK

installed (version 1.4 or later, available at the Sun web site).

The easiest way to create a Jwdp icon is to make a copy of the blue J icon and then edit the properties as follows:

1. edit shortcut target to end in j.jar rather than j.exe
2. Change Icon to select the jr.ico in the J directory

One reason for this icon to fail is that the file association between jar and java is broken. You can fix the association, or try a shortcut target like:

`c:\Program Files\Java\j2re1.4.0\bin\javaw.exe -jar c:\j501a\j.jar`

## Unix install

J is installed on your Unix system with file j501X\_Y.tar.gz (where X indicates a bug release level and Y is the uname in lowercase). Example unames are linux and darwin (for Mac OS X). You can download this file from [www.jssoftware.com](http://www.jssoftware.com).

The first step is to unpack the file to create a directory tree rooted at j501X. The file is a gzipped tar file and unpacking it varies on different systems. On Linux do the unpack with:

```
tar -xzf j501X_linux.tar.gz
```

The simplest install is to do the unpack in your home directory. If you do, you are ready to run J after the unpack.

Start Jwd with command `~/j501X/jw` and start Jconsole with `~/j501X/jc`. A common jw failure is that java isn't on your path and you can edit jw to give the full path.

Customize your installation by edits to jw and jc; copy them to bin directories; create links and icons.

There is a minimal man page for jconsole at `system/extras/help/man/jconsole.1` and you can make this available to man by copying the file to `/usr/local/man/man1` or other suitable directory.

You don't have to install in the j501X directory in your home directory, but installing in a different location requires extra steps. When you run J it needs to

find its library files and it looks first in ~/j501X. If you install J in another directory you must set an environment variable so that it can find the library files. The variable has the name JPATHj501X. For example, if you installed j501a to ~/programs, then you could edit the jw and jc files as follows:

```
#!/bin/sh
export JPATHj501a=~/programs/j501a
java -jar ~/programs/j501a/j.jar $*
```

```
#!/bin/sh
export JPATHj501a=~/programs/j501a
~/programs/j501a/jconsole $*
```

More complicated installations, for example installation in a shared, read-only directory, for use by multiple users are possible, and require both system admin and J programming knowledge. The profile.ijs has to be modified so that standard J user directories such as user and temp are properly set. For example, the following changes to profile.ijs would give each user their own temp and user directories in their ~/J directory:

```
USER_j_=: (2!:5'HOME'),'J'
TEMP_j_=: USER_j_
```

## Old Windows Release Notes (material not merged into general docs)

### J 4.06 Release

Data displayed with boxes is an important J feature. Previous versions of J for Windows, by default, used oem fonts with linedraw characters for boxes with solid lines. The alternative is `+-|` ascii characters. Linedraw boxes are pretty, but they are also a never ending nuisance. Oem fonts aren't ansi or unicode and are discouraged by Microsoft and other vendors. They aren't standard in Unix and J for Unix uses `+-|`. They are a problem in email. In theory html can handle oem fonts, but across platforms and browsers the reality is a nightmare. Jsoftware has decided that easy html documentation, email communication, and standard usage across platforms outweighs pretty boxes. If you want them for your own use, you can still have them.

Jsoftware favors ascii boxes and the default in this release is ansi "Courier New".

The online J books User, Primer, Phrases, Dictionary, and Release Notes are in html. Select menu Help|Help to get an overview of the J Help System and the integrated index.

A preliminary version of 'Learning J' by Roger Stokes and Ken Iverson's 'Computers and Mathematical Notation' are also available through the J Help System.

A new lab 'A J Introduction' by Ken Iverson is in the Languages category. There are five other new labs by Ken in the Live Texts category.

For details on new features in this release see the Release Notes in the J Help System. The following are a few topics covered in those notes.

By default, explicit definitions keep a copy of the original text with whitespace and comments. This is convenient for casual use and development, but can be a significant and unnecessary space overhead in large production systems. Foreign `9!:41` can change the default so that this space is not used. Production systems should startup with `9!:41[0`.

Locales are more efficient and limits have been removed.

Foreigns 9!:36 and 9!:37 provide output formatting control. Adjust output formatting with Edit|Configure|Parameters.

Dll callbacks are supported. See lab: DLL: Callback.

Symbol and unicode, significant new data types, have been added.

`try.` control structure has been extended and a new control word `throw.` has been added.

`assert.` is a new control word.

## **J 4.05 Release**

The prokey license was introduced in J 4.05

### **business**

We have consolidated under the name Jsoftware. Iverson Software Inc. has changed its name to Jsoftware Inc. and does the marketing and sales activities previously run by Strand Software (the Strand staff now work directly with Jsoftware).

### **documentation**

The Help for foreigners, wd commands, and runtime have not been updated. Use them in conjunction with the latest information here.

### **html help**

Help menu item HTML Help runs your browser on documents for new J features. Of particular interest are: Performance Monitor, Mapped Boxed Arrays, Special Code, and Execution Time Limit.

### **labs**

Ken Iverson's Lipshutz lab is a companion to the Seymore Lipshutz "Linear

Algebra" of Schaum's Outline Series. Chris Burke's Performance Monitor Utilities lab is an overview of the new performance monitor.

### **undo/cut/copy/paste (ctrl+zxcv)**

The old Win31 shortcuts (alt+backspace/shift+delete/ctrl+insert/shift+insert) are no longer supported in the session manager. This was necessary to simplify resolving conflicts between shortcuts for the session manager, forms, controls, and OCX.

### **debug**

dbr 2 ( 13!:0[2 ) does not require a prokey and records information before clearing the stack. The debug latent expression ( 13!:15 ) executes after the stack is cleared and debug is reset. See HTML Help for more information. The stack information is recorded as:

```
STACK_ERROR_INFO_base_=: (13!:11;13!:12;13!:13;18!:5) ''
```

### **jfiles**

A change in 3!:1 means that jfiles written with J version 405 cannot be read by earlier versions. J405 can read jfiles created by earlier versions. The change in 3!:1 supports writing and reading binaries in standard and reverse byte order. For more information, see Help|HTML Help|3!:1, 3!:2, and 3!:3 Extended.

### **regex and socket**

The regex (16!:x) and socket foreigners (17!:x) have been decommitted. regex.ijs and socket.ijs scripts now use dll call (cd) to provide the same services. This is similar to how we previously decommitted the data driver (14!:x) foreigners. This results in a smaller, more portable J engine and in more open systems where you have full access to the scripts and underlying system services.

Socket verbs return the result code linked with the result, rather than catenated as in previous releases.

### **command line**



The form is: [filename] [ /command [parameter] ] ..

Standard profile (system\extras\config\profile.ijs) is run unless there is a /noprofile command.

Filename is run after the standard profile.

There are 3 types of commands: windows, J startup, and application.

### **windows:**

/register (or /regserver) - register JEXEServer and JDLLServer  
/unregister (or /unregserver) - unregister JEXEServer and JDLLServer  
/embedding - start as Automation server for COM client

### **J startup (start with j, as will new ones)**

/noprofile - start without standard profile  
/jtemp path - temp directory  
/jddename servername  
/jrt - runtime application (see runtime section)

Application commands can be used in the application (wd'qmdline').

### **Examples:**

profile (show session)  
c:\j.exe  
profile (show session), run foo.ijs  
c:\j.exe user\foo.ijs  
profile (hide session), run foo.ijs (simple runtime app)  
c:\j.exe user\foo.ijs /jrt  
no profile, run foo.ijs (production runtime app)  
c:\j.exe app.ijs /noprofile /jrt

### **window driver**

wd'...' 65k argument limit removed and 65k result limit is now 500k.

makejr - decommitted (see runtime section)

makejl - decommitted (see runtime section)

picon filename n - set form icon with icon n from file (exe, dll, or ico file)

wd'picon system\examples\data\jy.ico 0' qrt - decommitted (see runtime section)

smicon filename n - set sm icon (see picon)

## **runtime**

The J license is free and you can distribute runtime applications that include as much of the J system as required.

Simple runtime applications can use the standard profile. Production runtime systems should be built with the Project Manager.

/rt parameter is decommitted. You have to rework old runtime applications.

/jrt parameter causes the standard profile to not do an smmfshow command to show the session manager. The runtime app can have the standard profile, but not show the session manager. If there is an error and the session manager is invisible, J terminates with a message box saying there was an error. If J finishes execution and the session manager is invisible and there are no forms, then J terminates.

wd'picon ...' sets form icons for the application.

wd commands makeijr and makeijl are desupported. ijr files are no longer supported. ijl files are created by 3!:6 .

system\examples\demo\runtime.ijs is a simple runtime application. Open and experiment with the script. Run it from an icon or Start|Run with command line:

```
c:\j405\j.exe system\examples\demo\runtime.ijs /jrt
```

system\examples\runtime\bldrt.bat creates a production runtime application setup and distribution. Study the bldrt.bat file and run it to create a distributable J runtime application.

## **ocx controls**

OCX controls with ids with blanks are supported by the form editor with quotes. For example: "ocx:rmocx.RealPlayer G2 Control.1"

OLE command picture and object arguments start with a !. For example:

```
wd'olemethod images listimages add ,, !picture:abc.bmp'
```

If the argument contains blanks, it must be quoted, but then it can't be distinguished from quoted data. A ! by itself is an escape so that the next parameter can be a quoted picture or object argument. For example:

```
wd'olemethod ... ,, ! "!picture:my abc file.bmp"
```

,

## **addons (LAPACK and FFTW)**

J addons are installed in the J addon directory. Old addon downloads won't install in the proper place for J4.05 and new downloads won't install in the proper place for previous releases.

## **COM**

The j.exe and j.dll tlb files are now included as resources and are no longer separate files. The J installation registers both the JEXEServer and JDLLServer COM objects. Use jreg.bat to register or unregister.

## **J 4.04 Release**

Release 4.04 has sparse arrays, changes in J COM objects, and OCX license support.

### **Sparse arrays**

Sparse arrays provide a compact and efficient storage form for very large arrays where most elements are zero or some other "sparse element". The sparse array representation does not store extra copies of the sparse element. J primitives work

directly on sparse arrays. A new verb \$. converts between sparse and dense representations of arrays.

In this release, only numeric arrays can be sparse. Subsequent releases will support sparse character and boxed arrays.

Run `lab Sparse Arrays` for a quick overview of the new facility. For detailed information, see `Help|HTML Help|Sparse Arrays`.

## **J COM objects**

`GetB` and `ErrorTextB` methods in previous versions incorrectly returned a `BSTR` with a count that included the terminating `NULL`. This bug has been fixed. Applications with workarounds may need fixing.

Multiple `JDLLServer` COM objects are now supported. Each `JDLLServer` object created is a complete new instance of `J`. Previously each `JDLLServer` object shared the same globals (for example, symbol tables) and was not useful. It is now possible to efficiently create completely independent, in-process, `J COM` objects. These independent `J COM` objects can run in the same threads or in different threads.

The direct DLL interface to `J` requires new, explicit calls to create and free a `J` instance and the instance handle is a new parameter to all other calls. For example:

```
HANDLE pj = JInit(); // get handle for new J instance
JDo(pj, "a=:i.5");
JFree(pj);           // free J instance
```

## **OCX license**

OCX controls distributed with a `J` application can be distributed in a runtime version that requires license information when they are created. On a system with the design time OCX installed, use `olegetlic` to get the license. The progid is the same id used in the `cc` command (without the `ocx:` prefix). To create the OCX on a system with a runtime OCX, use `olesetlic` to set the license. The following example uses the FarPoint spreadsheet control.

```
fpkey =: wd'olegetlic FPSpread.Spread.2'
fpkey
```

```
67 0 111 0 112 0 121 0 114 0 105 0 103 0 104 0 116 0 32 0 ...  
  (".fpkey) { a.    NB. 2 byte unicode is often readable  
Copyright ...
```

```
NB. in a distributed application set key before cc  
wd'olesetlic FPSpread.Spread.2 ',fpkey  
wd'cc ss ocx: FPSpread.Spread.2'
```

## **load utility**

The load utility has changed, when loading scripts from a directory that includes a project file. If the left argument of load is not given, then a default locale is used. Previously, this locale was always base; now, if the directory contains a project file, the default locale used is that specified in the projects target locale. This makes it easy to work with individual scripts in a project, since they will automatically load into the projects target locale.

## **Debug**

The Debug window now includes single-step and single-step-into buttons, plus several other commands available on shortcut keys.

The toolbar Open button now responds to the current cursor position. If the name at the cursor is a noun, its definition is displayed in a viewer; otherwise the script where the name is defined is opened.

See Debug for more details, or with the Debug window active, press Ctrl-H for a list of the available shortcut keys.

## **J 4.03 Release**

### **Code Editor**

The new code editor was written by Andrei Stcherbatchenko and we're sure you'll find it makes life as a J programmer more exciting and productive.

- customized code coloring (Edit|Configure)
- large files (up to 10,000 lines)

- multiple undo and redo
- drag and drop editing
- standard arrow/home/end shortcuts (ctrl /shift modifiers)
- Ins toggles overstrike/insert
- left margin
- quick line selection with mouse
- ctrl+shift+0-9 toggles mark
- alt+0-9 scrolls mark visible
- ctrl+shift+F2 clears marks
- Tab indents (shift+Tab exdents) selected lines
- fixed pitch font required
- ijk output truncated with . . . at 256 characters
- ctrl+shift+up/down arrow recalls lines (used to be ctrl+up/down)

## Form colors

You can set form and control (text, text background, and background) colors.

```
wd'pcolor R G B' NB. form color wd'setcolor id textR G B
textbkgnR G B bkgndR G B'
```

You can add these commands in form initialization.

```
abc_run=: 3 : 0
wd ABC
NB. initialize form here
wd 'pcolor 0 0 255'
wd 'setcolor ccreditm 255 0 0 0 0 255 0 0 255'
wd 'pshow;'
)
```

The setcolor command can override the gray readonly edit boxes. The setcolor command has no effect on push buttons or the dropdown listbox of a combobox (unfortunate Window facts).

## Debug

This release includes a preliminary version of a debug GUI. It is built on top of

facilities that have been in J for some time. This version is not complete, and requires changes in the J engine before it can provide all the facilities we envision. However, we found that it is already so useful that we decided to include it in this release. It is a taste of things to come.

Start Debug with Run|Debug. Press the Help button to see documentation. Learn about Debug with the Debug lab.

## **HTML Publish**

Use Run|HTML Publish, or Tools|HTML Publish from Project Manager, to convert scripts to HTML format for the web. Hold down the shift key to convert the active window.

## **Popup Menu**

The release includes Oleg Kobchenko's popup menu, as enhanced by Alex Kornilovski. For more information, see scripts system\packages\winapi\menu.ijs and menudemo.ijs.

```
load 'menu'  
wdmenu ' one two three'
```

## **ODBC**

This release decommits the Data Driver foreign family 14!:x . This interface was written in C and was a closed, black box to J programmers. It didn't support newer versions of the ODBC API and was missing important features.

The new ODBC support was written by John D. Baker and is provided by scripts that use DLL calls to directly access the ODBC API. The J programmer now has the same ODBC access as the C programmer. The new script dd.ijs provides the same functionality as the previous version with DLL calls to the ODBC API, rather than with 14!:x calls. An application that uses dd.ijs should work with the new dd.ijs. There are some performance improvements and significant new functionality such as support for stored procedures. If you are interested in ODBC, work your way through the new ODBC labs.

## **FTW AddOn**

In addition to the Lapack AddOn, there is now an FFTW AddOn. FFTW is a collection of fast C routines for computing the Discrete Fourier Transform in one or more dimensions. It includes complex, real, and parallel transforms, and can handle arbitrary array sizes efficiently. The FFTW AddOn consists of a DLL incorporating the FFTW routines, plus supporting J scripts and labs. The AddOn is currently only available for Windows 9x/NT. For more information and to download the FFTW AddOn, visit [www.jsoftware.com](http://www.jsoftware.com) (or use the J cdrom AddOn directory).

## **Project Manager (PM)**

There are several improvements in the Project Manager. If you're an experienced J user and you don't use PM, then we suggest it is time to start. It is great for both small and large projects. If you're new to J, start out right by using PM for all your projects. Start PM from menu command Run|Project Manager.

## **wd commands**

New wd commands (documented in Help|wd commands):

- `glgridspace x y` - space in from upper left for text
- `pcolor R G B` - form background color
- `setcolor id textR G B textbkgnR B G bkgndR B G` - control color
- `smcolor n R G B` - code editor color
- `smkeywords n keywords` - code editor keywords (for color)
- `smreplace text` - replace selected text
- `smgetscroll` - scroll position of top line
- `smflush [bool]` - force immediate (unbuffered) output

Enter in a readonly editm control is an enter event.

## **Miscellaneous**

File menu MRU (most recently used) files has 8 items and is retained between sessions.



Bug fixes (thanks to all you out there, particularly J forum members, for giving us the chance to fix them!).

```
try. catch. honors debug stops.
```

Several performance improvements in the J engine.

## **J 4.02 Release**

Try the new demos `pousse` and `eigenpictures` (J LAPACK AddOn is required for `eigenpictures`).

Take a look at the new lab `Fractals Visualization`, & `J`.

Run labs `Mapped Names and Files` and `Mapped File Database` to learn about these new facilities. A mapped file can be accessed as if it were memory and a mapped name is an array that is a file.

Run labs `DLL: Writing and Using a DLL` and `DLL: Using System DLLs` (file examples) to learn about changes to the 15!x DLL call facilities. The User Manual chapter `DLLs and Memory Management` has been updated to reflect the changes. Some of the changes are incompatible with previous releases. Previously this facility was only available in Win95 and NT. It will now be available on all platforms.

A J AddOn is a separately packaged installation that is added on to the base J installation.

The complete LAPACK library is now available with the J LAPACK AddOn. LAPACK (Linear Algebra Package) is a set of routines for solving systems of simultaneous linear equations, least-squares solutions of linear systems of equations, eigenvalue problems, and singular value problems. The associated matrix factorizations (LU, Cholesky, QR, SVD, Schur, generalized Schur) are also provided, as are related computations such as reordering of the Schur factorizations and estimating condition numbers.

Download `jlpack.exe` and run it after you have installed J4.02. This adds the LAPACK files to the J directory structure. When it is installed, run lab `LAPACK` to learn about this powerful new addition to J.

## J 4.02 Interpreter

The dyad `{.` has been extended to accept infinities in the left argument, with an infinite value specifying the length of the corresponding axis.

```
a=: i.4 5
a
0 1 2 3 4
5 6 7 8 9
10 11 12 13 14
15 16 17 18 19
  2 _ {. a          NB. first 2 rows and all columns
0 1 2 3 4
5 6 7 8 9
  _ 2 {. a          NB. all rows and first two columns
0 1
5 6
10 11
15 16
  _1 _ {. a          NB. last row and all columns
15 16 17 18 19
  _ {. a             NB. all rows and all columns
0 1 2 3 4
5 6 7 8 9
10 11 12 13 14
15 16 17 18 19
```

The monad `i:` has rank 0; `i: n` gives integers from -n to n inclusive; `i: a j. b` gives numbers from -a to a in b equal steps.

```
i: 5
_5 _4 _3 _2 _1 0 1 2 3 4 5
i: _5
5 4 3 2 1 0 _1 _2 _3 _4 _5
i: 5j4
_5 _2.5 0 2.5 5
i: _5j4
5 2.5 0 _2.5 _5
i: 2.5j4
_2.5 _1.25 0 1.25 2.5
i: 5j_4
|domain error
```

| i:5j\_4

On extended precision arguments, the determinant `-/.*` runs faster in less space.

```
ts=: 6!:2 , 7!:2@]          NB. time and space
ts '-/ .* h' [ h=: % >: +/~ i.10x
0.14 37888
ts '-/ .* h' [ h=: % >: +/~ i.20x
2.073 291520
```

## **J 4.01 Release**

Use `=:` for global definitions in scripts. `Run|Window` and `Run|File` use load and definitions made with `=:` are local to load and disappear when it finishes.

The J file suffix has changed from `.j?` to `.ij?` ( `.js` to `.ijs` ) to avoid javascript conflicts. Use `system\extras\migrate\ext.ijs` if you have lots of files.

system directory contains the other distributed directories (`main\stdlib.js` is now `system\main\stdlib.ijs`).

## **J User License (JUL)**

**You must have a JUL to use J as a programmer.**

A JUL is like a driver's license. It has an annual fee and expires if not renewed. There are 3 JUL types:

**Commercial** \$600 - you use J at work

**Non-commercial** \$100 - you do not use J at work

**Free** \$0 - students, new users, or you only use releases older than 18 months

You must renew your JUL annually at the then appropriate fee. If you stop using J, you can renew an expired JUL when you start again.

Users with a prokey valid for updates or covered by a site license still need a JUL and should indicate this status on an application that includes no payment.

Click on Email JUL to create an email message formatted with a JUL application.

If you prefer regular mail or want to pay by USD check or money order see [Mail JUL p7](#).

Teachers can apply for an Free JUL that covers their students, rather than having each student apply individually. Click on School JUL to create a formatted email message.

## **General Information**

[About J p5](#)

[Products p6](#)

[J User License p3](#)

[J User License Order Form p7](#)

[Support and Questions p8](#)

[Copyright / Warranty / License p9](#)

## About J

J is a general purpose programming language designed by Ken Iverson and Roger Hui. It is available on a wide variety of computers and operating systems. J is distinguished by its simple and consistent rules, a large set of built-in capabilities, powerful facilities for defining new operations, and a general and systematic treatment of arrays.

J Systems are developed and distributed by:

Jsoftware Inc.  
P.O. Box 330  
Excelsior, MN  
USA 55331

tel: 952 470-7345

fax: 952 470-9202

To get the latest information, see [www.jsoftware.com](http://www.jsoftware.com).

For sales, email: [sales@jsoftware.com](mailto:sales@jsoftware.com).

For general inquiries, email: [info@jsoftware.com](mailto:info@jsoftware.com).

For technical support, email: [tech@jsoftware.com](mailto:tech@jsoftware.com).

## Products

J is a high-level, general purpose programming language. The J system provides: an engine for executing J; various front ends that provide user interfaces to the J engine; a library, written in J, that provides an IDE (interactive development environment), numerous tools, utilities, demos, tutorials; and online documentation.

J Systems are available for Windows, Windows PocketPC, Macintosh, Linux, Solaris, AIX, FreeBSD, NetBSD, and others. The core language is identical in all versions, and programs not making use of platform-dependent features will work unchanged on all systems.

J is documented in 5 online html format books that are distributed with the system. These books, and others, are available in PDF format from the Jsoftware web site.

J is licensed for free installation on all platforms. You can download, redistribute, and run end user applications built with J for free.

You must have a [J User License \(JUL\) p3](#) to program in J.

## **J User License Application (New and Renewal)**

Print this page, complete the form, and send to:  
Jsoftware Inc., P.O. Box 330, Excelsior, MN 55331 USA

If there is no enclosure you can fax it to: 952 470-9202.

Checks must be in US dollars, or you can send international postal money orders in US dollars.

name:

email:

JUL (if this is a renewal):

address (used only by Jsoftware for direct mail):

JUL type: ☐ Commercial \$600 ☐ Non-commercial \$100 ☐ Educational \$0

Visa/MasterCard/American Express card #:

name on card:

expiry date:

card billing address (required - used only for credit card validation):



## Support and Questions

The J forum (a mailing list) is the best place for general questions on the J language. Visit [www.jsoftware.com](http://www.jsoftware.com) to join the forum and for the latest information and resources.

Check the online manuals and help files available through the J Help menu.

Send email inquiries to:

**general** [info@jsoftware.com](mailto:info@jsoftware.com)

**sales** [sales@jsoftware.com](mailto:sales@jsoftware.com)

**technical** [tech@jsoftware.com](mailto:tech@jsoftware.com)

When reporting problems, please include all relevant information.

## **Copyright / Warranty / License**

J Products are Copyright © 1994-2002 by Jsoftware Inc. All rights reserved.

### **Warranty and License Agreement**

Jsoftware Inc. ("Licensor") is willing to license the enclosed software to you only if you accept all of the terms in this license agreement. Please read the terms carefully before you install this package, because by installing the package you are agreeing to be bound by the terms of this agreement. If you do not agree to these terms, licensor will not license this software to you, and in that case you should delete the software and return any/all related written materials promptly, for a refund.

### **Ownership of the Software**

The software program, J ("Software") and the accompanying written materials are owned by Licensor [or its suppliers] and are protected by United States copyright laws, by laws of other nations, and by international treaties.

You may not reverse engineer, decompile, or disassemble the Software.

### **Limited Warranty**

Licensor warrants that the Software will perform substantially in accordance with the accompanying written materials for a period of 90 days from the date of your receipt of the Software. Any implied warranties on the Software are limited to 90 days. Some States do not allow limitations on duration of an implied warranty, so the above limitation may not apply to you.

Licensor disclaims all other warranties, either express or implied, including but not limited to implied warranties of merchantability, fitness for a particular purpose, and non-infringement, with respect to the software and the accompanying written materials. Licensor's entire liability and your exclusive remedy shall be, at licensor's choice, either (a) return of the price paid or (b) replacement of the software that does not meet licensor's limited warranty and which is returned to licensor with a copy of your receipt. Any replacement software will be warranted

for the remainder of the original warranty period or 30 days, whichever is longer. These remedies are not available outside the United States of America.

This limited warranty is void if failure of the software has resulted from modification, accident, abuse, or misapplication. In no event will licensor be liable to you for damages, including any loss of profits, lost savings, or other incidental or consequential damages arising out of your use or inability to use the software. If you have any questions concerning this agreement or wish to contact licensor for any reason, please write: Jsoftware Inc., P.O. Box 330, Excelsior, MN USA 55331 or call (952) 470-7345 or fax (952) 470-9202, email: [info@jsoftware.com](mailto:info@jsoftware.com).

U.S. Government Restricted Rights. The Software and documentation are provided with Restricted Rights. Use, duplication, or disclosure by the Government is subject to restrictions set forth in subparagraph (c)(1)(ii) of The Rights in Technical Data and Computer Software clause at DFARS 252.227-7013 or subparagraphs (c)(1) and (2) of Commercial Computer Software - Restricted Rights at 48 CFR 52.227-19, as applicable. Supplier is Jsoftware Inc.

## Starting J

Starting J in Windows and Unix is documented in [Release Highlights and Overview p1](#).

In addition the following Windows specific command line parameters are supported:

/ddename - set ddname on start up

/embedding - start as Automation server for COM client J startup

/register (or /regserver) - register JEXEServer and JDLLServer

/unregister (or /unregserver) - unregister JEXEServer and JDLLServer

[Mac J.402 Startup p11](#) for versions prior to the Unix based MAC OS X.

## **Mac Startup**

J requires a Macintosh with System 7 on a 68020 (or better) or a Power PC. Macintosh and Windows J for Macintosh is a port of J for Windows.

Create and install in a new folder, such as J4.

## **Help Files**

The J documentation is in Microsoft help files and the Microsoft folder contains the application that displays them.

The Microsoft folder contains files that are distributed with various systems and may already be on your system. Check if you already have a copy of the file Microsoft Help and check file dates to see if the diskette version is more recent. To install the diskette version, drag the Microsoft folder to the Extensions folder in your System Folder.

Double-click the J Dictionary icon and experiment with the help system.

## **J Font**

Drag JFontR to your System Folder and reply OK to put it in the Fonts folder. JFontR fonts have box drawing characters used by J.

## **Loading J**

Double-click the J icon to start J and run the standard profile script.

## **Menu shortcuts**

Menu shortcuts use the Command key where Windows uses the Ctrl key. The only difference is

Run/Window where Ctrl+W conflicted with Command+W for close.

Command+Y is Run/Window.

Command+W is close window (also for window driver windows).

Command+? opens J Dictionary help and displays the vocabulary.

F1 is the same as Command+? (Windows convention).

Ctrl+F1 opens J User help and displays the windows driver reference.

Ctrl+F6 cycles through session manager windows.

F2 through F9 are Tool menu shortcuts.

F10 enters a menu state similar to Windows. Press F10 and navigate the menu with arrows and letters. Select an item, press Esc, or click the mouse to exit this state.

### Icons

J program is a boxed black J.

J scripts are a boxed black J with a bent corner.

### Icon shortcuts

Double-click a script to run.

Ctrl+double-click a script to open.

### Window Driver

Window Driver windows are resizable but do not display to resize icon in the lower right corner. Resizable windows in Windows have a thicker frame and do not have a resize icon.

`glfloodfill` and `glchord` commands are not supported.

Ownerdraw buttons and listboxes are not supported.

Windows DIB (Device Independent Bitmap) files display in an ISPICTURE window. DIB files usually have a suffix of `.dib` or `.bmp`.

Mac PICT files display in an ISPICTURE window.

ODBC access is not supported.

### Floating point

J for the Mac (not the PowerPC) does not support the floating point coprocessor.

# Directory Paths in J

## Directory Foreigns

1!:43 y	Query Current Working Directory.
1!:44 y	Set Current Working Directory.
1!:45 y	Query Default Profile

See J Help on !: (foreign conjunctions) for further details.

## Directory Verbs

The J profile defines the following verbs in the z locale:

jssystempath	points to J system directory
jcwdpath	current working directory
jtemppath	temporary directory
juserpath	user directory
jconfigpath	user configuration directory
jaddonspath	addons directory

These verbs are used to provide the full pathname for files, and to ensure that the path separator is correct for the host operating system. For example:

```
jssystempath 'system\main\dates.ijs'
c:\j5\system\main\dates.ijs
```

```
jconfigpath 'startup.ijs'
c:\j5\user\config\startup.ijs
```

```
jtemppath''
c:\j5\temp
```

Note that jssystempath *points* to the J system directory, while other verbs *include* their directories.

The verbs, other than `jcwdpath`, are created from corresponding nouns defined in the `j` locale, and specified in the J profile script (`profile.ijs`), namely `SYSTEM`, `TEMP`, `ADDON`, `USER` and `CONFIG`. For example:  

```
18!:4 <'j' system=: profile  
{.~ profile i: pathsep user=: system,pathsep,'user'
```

These nouns are only used when creating the directory verbs. They are not referenced after J has been loaded and the profile has been run. The directory verbs are created before running the user's Startup script.

## User Configuration

You can configure the directory verbs by changing the nouns defined in `profile.ijs`. Otherwise, once the profile has been run, you can configure these verbs only by redefining them - changing the corresponding nouns has no effect. In particular, you would need to redefine the directory verbs if you wanted to configure them in your Startup script.



## **Session Manager**

[Session Manager p14](#)

[Execution Windows p15](#)

[Script Windows p16](#)

[Input Log p17](#)

[Menus p18](#)

[Status Bar p19](#)

[Controlling p20](#)

## Session Manager

J for Windows combines the language interpreter with a user interface called the *session manager*.

The session manager is a Windows multiple document interface (MDI) application, which allows the use of several document windows at one time within the same session. The session manager has a header, a menu line, Status Bar (which may be turned off), and one or more document windows.

You use the session manager by working with the document windows. You can have as many of these windows open as you wish - each window is attached to the same session. Only one window is active at a time.

There are two types of document window: execution windows and script windows:

- An *execution window* allows sentences to be evaluated. When you type into the execution window and press Enter, the sentence you entered is executed, and the result is displayed below.

- A *script window* allows sentences to be entered without being evaluated. Typically you use script windows to write your applications, then run the scripts by loading them into an execution window.

Both window types represent ordinary files that can be saved during the session for use later on. Both are Windows edit controls, and you can use standard Windows commands to manipulate them, for example you can cut and paste between the windows and the Clipboard. You can move them about, resize them and minimize or maximize them.

## Execution Windows

Execution windows are distinguished by their file extension of .ijx. When you load J, it creates a new execution window for the session. These windows are named 1.iyx, 2.iyx etc. By default, 1.iyx is used, however, if a file of this name already exists, the next available unused name is picked.

You can type sentences into an execution window. When you press Enter, the system reads the line on which the cursor is positioned. If this line is at the foot of the execution window, it is executed and the result displayed below. Otherwise, the line is copied to the foot of the execution window - press Enter again to execute it.

There is always at least one execution window, but you can open as many as you wish - all are attached to the same session. This is useful for experimenting with some sentences - you could open a new execution window to do so, without writing to your original execution window. Open a new execution window by selecting New IJX from the File menu.

The execution windows 1.iyx, 2.iyx and so on represent temporary files. That is, while the J session is active, if you have a window 2.iyx, then there is a file of that name (temp\2.iyx), but when you close the window or terminate the J session, this file is deleted. If you want to save these files, you must explicitly save them with a name other than n.iyx (for integer n); for example, mywork.iyx. If you do save an execution window in this way, the session manager treats it as a permanent file, and will save the file again when you close the window or terminate the session. You can load it in your next session - it will again be treated as a permanent file.

In no case are you prompted when closing an execution window - if it is temporary, the file is deleted; if it is permanent, the file is saved.

Note that you cannot close the original execution window.

## Script Windows

Script windows are typically distinguished by their file extension of .ijs. Note that this extension is not required - in fact any file that does not have an extension of .ijx is treated as a script window. However, it is good practice to use .ijs for any file intended as a J script.

When you create new script windows, the names used are 1.ijs, 2.ijs and so on. By default, 1.ijs is used, however if a file of this name already exists, the next available unused name is picked.

You can type sentences into a script window, but these are not evaluated. When you press Enter, the cursor is simply moved to the line below. To run a script window, you can either select an execution window and then enter a sentence to load the corresponding script file, or you can run it directly by selecting one of the options from the Run menu or pressing the equivalent Ctrl key:

Window	Ctrl+W	runs script window
Selection	Ctrl+E	runs selected text only
Line	Ctrl+R	runs current line only
File	Ctrl+T	runs a script file, selecting from the File/Open dialog box

By default, these options are run silently - script output will only display if there is

an error. To run the options with display on, hold down the Shift key when you click on the menu option, or press Shift-Ctrl-key.

When you select the Run/Window menu option, the session manager first saves the script as a file (if changes have been made), then loads this file into the most recently active execution window. In some cases, you do not have to switch to the execution window at all - for example if you are developing a Windows application, then you can create and test the parent and child controls directly from the script window.

You can open as many script windows as you wish - all are attached to the same session.

The script windows 1.ijs, 2.ijs and so on represent temporary files. That is, while the session is active, if you have a window 2.ijs then there is a file of that name (temp\2.ijs), but when you close the window or terminate the session, you are prompted for the file to be deleted. You can save these files if you wish with the same name, but when you load them again, they are still treated as temporary and you will again be prompted to delete them when they are closed.

If you explicitly save them with a name other than n.ijs, for example, mywork.ijs, then the J session manager treats them as permanent files. If you close a permanent file or terminate the J session, you are prompted to save the file, if it has been changed. You are also prompted to save the file when you first save changes to a permanent script file that you have loaded in a session (which may be when you run it as a script).

Prompting for script windows is therefore as follows:

- if the file is temporary (i.e. the name is n.ijs), you are prompted to delete the file when it is closed.
- if the file is permanent (any name other than n.ijs) and you have made changes to it, you are prompted to save the file when it is closed, or when you first run it as a script.

## Input Log

All sentences entered into execution windows are stored in an input log. You can recall entries by either pressing the Ctrl up-arrow and Ctrl down-arrow keys to cycle backwards and forwards through the log; or by selecting an entry from the Input Log listbox obtained from menu Edit|Input Log or by pressing Ctrl+D.

When you recall an entry from the input log, it is read into the currently active execution window, or the most recently active execution window if the currently active window is a script window. Only one input log is maintained, even if there is more than one execution window.

The input log does not store duplicate entries. If you recall an item from the middle of the log and execute it, the item now only appears at the end of the log - the entry in the middle of the log is deleted.

## Menus

The session manager includes a Menu bar which for the most part, contains standard Windows menus:

The **File** menu is used for file access: open, close, save, delete, print. It also includes a list of the most recently accessed files. You can also exit the session by selecting: File, Exit.

The **Edit** menu provides standard edit capabilities, Undo, Cut, Copy and Paste. You can also use this menu to restore a window from file (overwriting changes made since the file was originally loaded), or to toggle a file's Read Only status. A file marked Read Only may not be changed, until its Read Only status is explicitly removed.

Edit|Form Edit... loads the Form Editor

Edit|Project Manager... loads the Project Manager

Edit|Configure... loads the session configuration dialog.

The **Run** menu allows you to run a script window, highlighted text, the current line, or a script file, either silently or, if the Shift key is held down, with output displayed in an execution window. You can also set the locale into which a script is loaded.

The **Tools** menu contains user-defined items. This menu can be customized as described in the section below on Session Manager Commands.

The **Studio** menu lets you run the Labs and Demos.

The **Window** menu allows standard window positioning (e.g. Tile, Cascade), and allows you to select a window as the current active window. (You can also select the active window by pressing Ctrl-F6 to cycle through the active windows.)

The **Help** menu provides help facilities, including access to the full text of the J manuals.

## Status Bar

The Status Bar is shown at the foot of the screen. It displays a help message, followed by several status boxes:

- Ready/Running shows whether the interpreter is waiting for input, or running the previous input.
- CAPS shows the status of Caps Lock
- NUM shows the status of Num Lock
- The two numbers (e.g. 00011/0021) show the cursor position in the active window.

You can hide the Status Bar by unchecking the Status Bar item in the Edit|Configure... dialog.



## Controlling the Session Manager

A J program can use the Window Driver `wd` to access some aspects of the Session Manager, such as the Tools menu document windows. The names of Window Driver commands specific to the session manager start with the letters `sm`.

### Tools Menu

You can add your own menu items to the Tools menu using `smsetcmd`. These also define the function keys. Your menu items can be accessed either by selecting the item directly from the Tools menu, or by pressing the corresponding function key. Function keys F2-F9 are available for this purpose.

The form is:

```
wd 'smsetcmd num type name sentence;'
```

where:

- `num` corresponds to the function key being defined.
- `type` is one of :
  - 0 = remove definition
  - 1 = add definition, when invoked displays the sentence being run.
  - 2 = add definition, when invoked does not display the sentence being run.
- `name` is the text that appears on the Tools menu
- `sentence` is the sentence that will be run when the menu item is invoked.

For example, the following will assign the sentence `load 'laserjet'` to the function key F2 and corresponding menu item LaserJet. The sentence will display when invoked:

```
wd 'smsetcmd 2 1 "LaserJet" "load ''laserjet''";'
```

When you next select the Tools menu, you will see the new menu item LaserJet. It is a good idea to define a "HotKey" letter and include the number of the corresponding function key in the menu item. If you do this, you can also right-justify the function key name by preceding it with a TAB. Thus, the above command could have been entered as:

```
wd 'smsetcmd 2 1 "&LaserJet',TAB,'F2" "load ''laserjet''";'
```

The menu item will now appear as:

```
LaserJet F2
```

To remove this definition, enter:

```
wd 'smsetcmd 2 0;'
```

## Document Windows

There are several Window Driver commands to access the document windows. In most cases, you must first select the window that is to be the target of subsequent commands (this need not be the active window). To select a window, you actually select its filename:

qsmact	file name of selected window
qsmall	file names of all windows
qsmcsize	size of current execution window
qsmout	file name of current execution window
qsmsize	size of selected window (pixels)
qsmwh	size of MDI client area (pixels)
smappend	appends text to selected window
smcascade	cascade windows
smclose	close selected window

<code>smfocus</code>	activates selected window
<code>smmove xywh</code>	move and size selected window (pixels)
<code>smopen</code>	opens selected filename, if not already open
<code>smread</code>	reads data from selected window
<code>smsave</code>	saves selected window (if modified)
<code>smscroll n</code>	scrolls selected window
<code>smsetselect</code>	set selected text
<code>smssel</code>	selects window, by its filename
<code>smshow param</code>	shows selected window, with parameters from the set:
<code>sw_hide</code>	<code>sw_maximize</code>
<code>sw_minimize</code>	<code>sw_restore</code>
<code>sw_show</code>	<code>sw_showmaximized</code>
<code>sw_showminimized</code>	
<code>sw_showminnoactive</code>	
<code>sw_showna</code>	<code>sw_shownoactivate</code>
<code>sw_shownormal</code>	
<code>smsetcmd n t name sentence</code>	modifies Tools menu
<code>smtile</code>	tile windows down
<code>smtilea</code>	tile windows across
<code>smwrite</code>	writes text to selected window

For example, the following will open file `user\mywork.ijs` for editing and make it the active window:

```
wd 'smssel "user\mywork.ijs";smopen;smfocus'
```

Having opened the window, you could write some text (e.g. in the noun `TXT`) to it as follows:

```
wd 'smwrite *',TXT
```

## **Script Libraries**

[Directory Layout p29](#)

[Script Library Overview p30](#)

[scriptdoc utility p31](#)

[Definition Summaries p80](#)

[Definitions by Script p81](#)

## **system\main**

The is the main script directory.

### **Lib scripts**

Scripts with "lib" in the name are automatically loaded by the standard profile, In distributing scripts, you can assume that these definitions are available. In addition, these scripts, except for jadelib, are included in any projects built by the Project Manager. jadelib is an exception since it is has facilities intended to support development, not runtime applications.

colib

class/object library (z locale)

jadelib

Standard jade library (j locale)

loadlib

script load library (j locale)

stdlib

standard J library (z locale)

winlib

standard windows library (z locale)

### **Other scripts**

compare

comparison utilities

convert

conversion utilities

dates

date and time utilities

dd

data driver utilities

debug

debug definitions and utilities

dir

directory utilities

dll

utilities for calling DLLs

files

file access utilities

format

formatting utilities

gl2

2D graphics definitions

gl3

3D graphics definitions

jmf

mapped files

misc

miscellaneous utilities

myutil

example scripts for user definitions

numeric

numeric utilities

pack

package utilities

parts

partition functions

regex

Regular expression pattern matching

socket

utilities for using sockets

strings

string manipulation

text

text utilities

trig

trigonometric functions

validate

data validation



## **system\packages**

This directory contains various utility packages.

autocode\grid

grid autocode

color\cfmt

color formatter

color\cfmtfns

color fmt functions

color\color16

table of 16 primary colors supported by HTML

color\colortab

colortable used in J (from Internet Explorer)

color\ns216

Netscape primary colors (256 color table)

color\ntcolor

name that color (displays color table)

color\rgb

convert between color triples and RGB values

color\wdcolor

color table for wd 'qcolor'

color\xwin

X Windows color table

dde\server1

Sets up J as a hot-link DDE server.

dde\server2

Sets up J as a cold-link DDE server.

files\csv

CSV file access utilities

files\dbase

dBASE file access utilities

files\jfiles

Component File definitions

files\keyfiles

Keyed-file definitions

files\kfiles

Older keyed-file definitions (superceded by keyfiles.ijs)

files\nfiles

file read/write in various formats

finance\actfns

actuarial functions

finance\actutil

actuarial misc utilities

finance\interest

interest functions

graphics\bmp

supports bitmap files

graphics\gnuplot

supports gnuplot for Windows

graphics\gputil

gnuplot utility functions

graphics\isigmain

supports graphics using the isigraph control

graphics\isigraph

loads all isigraph files

graphics\isigutil

isigraph utilities

graphics\vkeys

vkey definitions

math\bigpi

calculate several digits of pi

math\brent

Brent's method in J

`math\det`

definitions for determinants

`math\fermat`

fermat - find factor of n near square root

`math\gcd`

calculate GCD

`math\integer`

various integer definitions

`math\integrat`

various methods for numeric integration

`math\jacobi`

jacobi's method for eigenvalues and vectors

`math\legendre`

legendre symbol and quadratic residues

`math\linear`

solve linear equations

`math\makemat`

make various matrices

`math\matfacto`

matrix factorization

math\mathutil

math utilities

math\matutil

matrix utilities

math\mean

various means

math\numbers

various number definitions

math\pollard

Pollard factorizations

math\poly

polynomial functions

math\primutil

prime testing programs

math\quatern

definitions for quaternions

math\spline

calculate cubic spline

math\svd

singular value decomposition

misc\datefmt

date formatters

misc\evolute

calculate evolutes

misc\fgh

displays calling sequence for J expressions

misc\fields

symbolic field indexes and access functions.

misc\ieee64

IEEE 64 bit representation of real number

misc\primitiv

names for primitives

misc\rtfview

view rtf text in richedit control

misc\telecom

serial port I/O API cover functions

misc\vkeyshow

shows vkey names

misc\xenos

suggested names for external conjunctions

ocx\graph

GraphLib definitions

ocx\ocxutil

Utilities for OCX controls

print\label

print labels

print\laserjet

LaserJet utilities

print\print

print package

print\prntn

prints text in n columns on a page

print\write

write package

regex\regbuild

utilities for building regular expressions

regex\regdemo

regular expression demo

regex\regj

nouns for applying regular expressions to J code

regex\regjbl

build regular expressions for J code

stats\random

various random utilities

stats\statdist

Statistical Distributions

stats\statfns

Statistical functions

stats\stats

load statistics files



## **system\classes**

This directory contains definitions for J classes.

grid\jfgrid

jfgrid class

grid\jfreport

provides jfreport form

grid\jtable

provides jtable form

grid\jtgrid

jtable grid classe

grid\jwatch

watch form

grid\jwgrid

provides watch grid for data

grid\jzgrid

base grid class

input\jinput

provides jinput form

opengl\jzopengl

base opengl class

opengl\opengl

loads the jzopengl class

pbc\pbc

example class: baggage check

pbc\pbcx

example extend class baggage check

plot\jzplot

base plot class

plot\plot

session definitions for plot

plot\plotdefs

plot defaults

## **system\extras**

This directory is used for system facilities, such as labs, configuration and help files. The layout is:

extras\config

configuration files

extras\help

help files

extras\java

Java interface support

extras\labs

labs

extras\migrate

migration aids

extras\order

order form

extras\template

template files

extras\util

utilities

## **system\examples**

This directory contains the demos and examples that are available from the menu Studio|Demos....

**user**

This is an empty directory set up as a convenience for the user's own scripts. The layout is:

classes

class definitions

config

configuration

projects

project files

**temp**

This is the default directory used for temporary ijs and ijs files created by the J system.

You can choose another directory for this purpose using the /temp start up parameter. This would be useful if J were installed on a LAN where you do not have write access.

## Directory Layout

The J executables j.exe and j.dll are installed in the main J directory, for example, c:\j406.

The following system subdirectories are created by the J installation:

System\Main

Main libraries and utilities

System\Packages

Packages (e.g. Stats, Printing)

System\Classes

Class definitions (e.g. Grid, Plot)

System\Extras

Various system utilities

System\Examples

Various Examples

In addition, two other subdirectories are created when J is installed:

User

User Definitions

Temp

Temporary files

Subdirectory Addons is created when one of the J Addons is installed.



## **Script Library Overview**

[system\main p22](#)

[system\packages p23](#)

[system\classes p24](#)

[system\extras p25](#)

[system\examples p26](#)

[user p27](#)

[temp p28](#)

## Script Documentation

System scripts are documented in comments in the scripts themselves. The scriptdoc utility can be used to view the documentation. To load enter:

```
load 'scriptdoc'
```

To view script documentation, e.g. for dates and data driver, enter:

```
scriptdoc 'dates dd'
```

### Documentation Conventions

The following conventions are used to document J system scripts, and are recognized by the scriptdoc utility.

Scripts are documented in comment lines (i.e. lines beginning with NB.). Multi-line comments are consecutive lines beginning with NB. with no break. The first line of a multi-line comment is used for summary documentation.

The script itself is documented at the beginning. The first line describes the script, i.e

```
NB. script description...
```

Definitions produced by the script are documented in comments where the first comment line begins NB.\* and has the form:

```
NB.*name type description
```

where type is a single letter:

a     adverb  
c     conjunction  
n     noun  
v     verb

The rest of a multi-line comment for a definition is free form, however, conventionally it may include lines beginning:

form:         describes the usage of the definition, e.g. NB. form:   calendar  
              year [ ,months]

example:     example, e.g.

NB. example:  
NB. todate 72460  
NB. 1998 5 23

Other NB. comment lines in the script are ignored for documentation purposes.

**bmp: supports bitmap files**

Name	Type	Description
hue	verb	generate color from color set
readbmp	verb	read bitmap file, returning RGB data
readbmphdr	verb	read header from bitmap file
viewbmp	verb	bitmap viewer using isipicture control
viewrgb	verb	view numeric matrix of RGB data
writebmp	verb	write bmp file from RGB data

**hue**

generate color from color set

x. is color set

y. is values from 0 to 1, selecting color

**readbmp**

read bitmap file, returning RGB data

new version thanks to Andrew Nikitin (J Forum 5 Sep 2002)

**readbmphdr**

read header from bitmap file

returns: bitsize, rows, columns

**viewbmp**

bitmap viewer using isipicture control

```
x. = [parentid [;parentname]]
```

```
y. = filename(s)
```

```
e.g. viewbmp jsystempath 'system\examples\data\j.bmp'
```

**viewrgb**

view numeric matrix of RGB data

(writes bitmap file, then views it)

x. = [parent]

y. = matrix

**writebmp**

write bmp file from RGB data

Form: data writebmp filename [;minimum bitsize]

picks appropriate bit size of 4 8 or 24, subject to optional minimum bit size.

**colib: class/object library**

Name	Type	Description
coclass	verb	set current co class
cocreate	verb	create object
cocurrent	verb	set current locale
codestroy	verb	destroy current object
coerase	verb	erase object
coextend	verb	add class locale to path (before z)
cofullname	verb	return name with locale qualifier
coinsert	verb	insert into path (before z)
coname	verb	return current co name
conames	verb	formatted co name list
conew	verb	create object
conl	verb	return co name list
copath	verb	set/get co path
coreset	verb	destroy object locales,

**coerase**

erase object

example: coerase <'jzplot'

**coextend**

add class locale to path (before z)

appends to COCLASSPATH

example:

```
coextend 'cdir'
```

```
coextend 'cdir pobj'
```

```
coextend 'cdir';'pobj'
```

**coinsert**

insert into path (before z)  
coinsert 'cdir'  
coinsert 'cdir pobj'  
coinsert 'cdir';'pobj'

**conl**

return co name list  
form: conl n  
0 e. n = return named locales  
1 e. n = return numbered locales  
conl '' = return both, same as conl 0 1

**coreset**

destroy object locales,  
other than for jijs and jijx windows

**color16: table of 16 primary colors supported by HTML**

Name	Type	Description
COLOR16	noun	table of HTML primary colors



**colortab: main colortable used in J**

Name	Type	Description
COLORTABLE	noun	main colortable

## compare: comparison utilities

Name	Type	Description
compare	verb	compare character data
fcompare	verb	compare two text files

### compare

compare character data

returns list of differences.

arguments may be character strings, with lines delimited by LF, or character matrices (trailing blanks ignored).

for Mac, tolerates lines delimited by CR.

result shows lines not matched, in form:

n [l] text

where:

n = 0=left argument, 1=right argument

[l] = line number

text = text on line

omits comparison of swapped lines where

1e7 <: product of number of non-trivial lines

### fcompare

compare two text files

form: opt fcompare files

opt is optional suffix

files is 2 file names or prefixes

example:

```
fcompare jsystempath 'system\main\myutil.ijs
\jbak\system\main\myutil.ijs'
'\myutil.ijs' fcompare jsystempath 'system\main
\jbak\system\main'
```

**convert: conversion utilities**

Name	Type	Description
av	verb	convert between characters and indices
detab	verb	remove tab stops
dfh	verb	decimal from hex
hex	adverb	create verb for hex calculation
hfd	verb	hex from decimal
mfv	verb	matrix from vector
quote	verb	quote text
vfm	verb	vector from matrix

**av**

convert between characters and indices  
 e.g.   av 'abcde'

**detab**

remove tab stops  
 remove tabs from character string  
 left argument is tab width, default 4

**hex**

create verb for hex calculation  
 e.g. 'FF' + hex '8'

**mfv**

matrix from vector  
 [delimiter] mfv vector  
 default delimiter is ' '

**quote**

quote text  
quote 'Pete''s Place'

### **vfm**

vector from matrix  
vector from matrix, lines separated by LF

**coutil: class/object utilities**

Name	Type	Description
cofind	verb	find objects containing name in object:
coinfo	verb	return info on co classes
conouns	verb	nouns referencing object
conounsx	verb	object references with locales
copathnl	verb	path name list
copathnlx	verb	formatted path name list with defining classes
coselect	verb	select current locale
costate	verb	state of class objects

**coinfo**

return info on co classes

returns noun refs;object;creator;path

**conounsx**

object references with locales

returns: object;references;locales

**copathnl**

path name list

path name list

**coselect**

select current locale

requires wdselect

**csv: read/write comma-separated value data (\*.csv) files**

Name	Type	Description
readcsv	verb	reads csv file into a boxed array
writcsv	verb	writes a boxed array to a csv file



## **getdate**

get date from character string

form: [opt] getdate string

useful for input forms that have a date entry field

date forms permitted:

1986 5 23

May 23 1986

23 May 1986

and:

opt=0 (days first - default)

23 5 1986

opt=1 (months first)

5 23 1986

other characters allowed: , - / :

if not given, century defaults to current

only first 3 characters of month are tested.

examples: 23/5/86; may 23, 1986; 1986-5-23

requires: valdate

## **isotimestamp**

format time stamps as: 2000-05-23 16:06:39.268

y. is one or more time stamps in 6!:0 format

## **timestamp**

format time stamps as: 23 May 1998 16:06:39

y. is time stamp, if empty default to current time

## **to date**

converts day numbers to dates

converts day numbers to dates, converse <todayno>

This conversion is exact and provides a means of performing exact date arithmetic.

y. = day numbers



x. = optional:  
0 - result in form <yyyy mm dd> (default)  
1 - result in form <yyyymmdd>

example:  
todate 72460  
1998 5 23

todate 0 1 2 3 + todayno 1992 2 27  
1992 2 27  
1992 2 28  
1992 2 29  
1992 3 1

## **todayno**

converts dates to day numbers  
converts dates to day numbers, converse <todate>

y. = dates  
x. = optional:  
0 - dates in form <yyyy mm dd> (default)  
1 - dates in form <yyyymmdd>  
0 = todayno 1800 1 1, or earlier

example:  
todayno 1998 5 23  
72460

## **tsdiff**

differences between pairs of dates.

form: end tsdiff begin  
end, begin are vectors or matrices in form YYYY MM DD  
end dates should be later than begin dates

method is to subtract dates on a calendar basis to determine  
integral number of months plus the exact number of days  
remaining.

This is converted to payment periods, where # days remaining are  
calculated as: (# days)%365

example:  
1994 10 1 tsdiff 1986 5 23  
8.35799

**tsrep**

timestamp representation as a single number

form: [opt] timerep times

opt=0 convert timestamps to numbers (default)

1 convert numbers to timestamps

timestamps are in 6!:0 format, or matrix of same.

examples:

tsrep 1800 1 1 0 0 0

0

"!:1.13 tsrep 1995 5 23 10 24 57.24  
6165887097240

**valdate**

validate dates

form: valdate dates

dates is 3-element vector

or 3-column matrix

in form YYYY MM DD

returns 1 if valid

**weekday**

returns weekday from date, 0=Sunday ... 6=Saturday

arguments as for <todayno>, e.g.

5 = weekday 1997 5 23 = 1 weekday 19970523

**dd: data driver utilities**

Name	Type	Description
ddcheck	verb	check response, display any error message
ddcnm	verb	column names selected by ddsel
ddcnt	verb	rowcount of last ddsq command
ddcol	verb	column names in the table
ddcom	verb	commit transaction
ddcon	verb	connect to data source
dddis	verb	disconnect from data source
ddend	verb	end sql statement started with ddsel
dderr	verb	error info on last command (name, source, warning, msg)
ddfch	verb	as ddfet, but with data in columns
ddfet	verb	fetch next record from selected data
ddrbk	verb	rollback transaction
ddsel	verb	prepare and execute sql statement (selection)
ddsql	verb	prepare, execute, and end sql statement
ddsrc	verb	data source names available from ODBC manager
ddtbl	verb	selection handle for tables in data source
ddtrn	verb	begin transaction

**ddcheck**

check response, display any error message

example: `ch=: ddcheck ddcon 'dsn=Access97'`

**ddcnt**

rowcount of last ddsq command

utility:



**debug: debug definitions and utilities**

Name	Type	Description
dberm	verb	last error message
dberr	verb	last error number
dbg	verb	turn debugging window on/off
dbjmp	verb	jump to line number
dblocals	verb	display names and locals on stack
dblqx	verb	latent expression query
dblxs	verb	latent expression set
dbnxt	verb	run next (skip line and run)
dbq	verb	queries suspension mode (set by dbr)
dbr	verb	reset, set suspension mode (0=disable, 1=enable)
dbret	verb	exit and return argument
dbrr	verb	re-run with specified arguments
dbrrx	verb	re-run with specified executed arguments
dbrun	verb	run again (from current stop)
dbx	verb	display stack
dbsig	verb	signal error
dbsq	verb	stop query
dbss	verb	stop set
dbst	verb	returns stack text
dbstack	verb	displays call stack with header
dbstk	verb	call stack
dbstop	verb	set stops on all lines in namelist
dbstopme	verb	set stops on current definition if y.
dbtrace	verb	trace control

## **dblocals**

display names and locals on stack

form: [namelist] dblocals stack indices (default all)

example:

```
        dblocals ''          display all local names in stack
'abc Z' dblocals i.5        display names abc and Z where found
                             in first 5 definitions on stack
```

## **dbstack**

displays call stack with header

if x.=0 ignores definition and source script (default)

1 displays full stack

y. is the number of lines to display, all if empty

limits display of &.> item to length 30.

## **dbstopme**

set stops on current definition if y.

does nothing if suspension is off

**dir: directory utilities**

Name	Type	Description
dir	verb	directory listings
dircompare	verb	compare files in directories
dirfind	verb	find name in directory
dirpath	verb	directory paths
dirs	verb	browse files in directory
dirss	verb	directory string search
dirssrplc	verb	directory string search and replace
dirtree	verb	get filenames in directory tree
dirused	verb	get count and space of files in directory tree

**dir**

directory listings

```
y. = dos file specification:
    if empty, defaults to *

x. is optional:
  - if not given, defaults to 'n'
  - if character, returns a formatted directory,
    where x. is the sort option:
      d=by date
      n=by name
      s=by size
  - if numeric, there are 1 or 2 elements:
      0{ 0= list short names
          1= boxed list of full pathnames
          2= full directory list
      1{ 0= filenames only (default)
          1= include subdirectories
```

subdirectories are shown first  
filenames are returned in lower case

e.g. `dir ''`  
1 `dir jsystempath 'system\main\d*.ijs'`

## **dircompare**

compare files in directories

form: `[opt] dircompare dirs`

`dirs` = directory names  
`opt` is optional, with up to three elements:  
0{ =0 short file comparison (default)  
    =1 long file comparison  
1{ =0 given directory only (default)  
    =1 recurse through subdirectories  
2{ =0 file contents only (default)  
    =1 also compare timestamps

e.g. `dircompare 'main \jbak\main'`

returns text result or error message

## **dirfind**

find name in directory

find name in directory

form: `string dirfind directory`

returns filenames in directory tree containing string

e.g. `'jfile' dirfind 'packages'`

## **dirpath**

directory paths

return directory paths starting from `y`.

optional `x.=0` all paths (default)

1 non-empty paths (i.e. having files)

e.g. `dirpath 'examples'`

## **dirs**

browse files in directory

e.g. `dirs jsystempath 'system\main\*.ijs'`



## **dirss**

directory string search

form: string dirss directory

searches for files in directory tree containing string,  
returning formatted display where found.

e.g. 'create' dirss 'main'

If x. is a 2-element boxed list, calls dirssrplc

## **dirssrplc**

directory string search and replace

form: (old;new) dirssrplc files

example:

('old';'new') dirssrplc jsystempath 'system\main\\*.ijs'

## **dirtree**

get filenames in directory tree

return filenames in directory tree as boxed matrix

optional x. is a timestamp to exclude earlier files

each row contains: filename;timestamp;size

e.g. dirtree ''

dirtree 'main'

dirtree jsystempath 'system\packages\\*.ijs'

1997 5 23 dirtree '' - files dated on or after date.

directory search is recursive through subdirectories

filenames are returned in lower case

**dll: utilities for calling DLLs**

Name	Type	Description
AND	verb	bitwise AND (&)
bitwise	adverb	bitwise operations
cd	verb	call DLL procedure
cdcb	verb	callback address
cder	verb	error information
cderx	verb	GetLastError information
cdf	verb	free DLLs
fc	verb	float conversion
fh	verb	free header
gh	verb	allocate header
ic	verb	integer conversion
mema	verb	memory allocate
memf	verb	memory free
memr	verb	memory read
memw	verb	memory write
OR	verb	bitwise OR ( )
symget	verb	get address of locale entry for name
symget	verb	set array as address
XOR	verb	bitwise XOR (^)

**bitwise**

bitwise operations

(monadic and dyadic)

e.g.  $7 = 1 \text{ OR } 2 \text{ OR } 4$   
 $\quad = \text{OR } 1 \ 2 \ 4$

**ic**

integer conversion

conversions

e.g.

```
25185 25699  =  _1 ic 'abcd'
'abcd'  =  1 ic _1 ic 'abcd'
1684234849 1751606885  = _2 ic 'abcdefgh'
'abcdefgh'  =  2 ic _2 ic 'abcdefgh'
```

**symget**

set array as address

15!:6 - get address of locale entry for name

15!:7 - set array as address

15!:8 - allocate header

15!:9 - free header

**files: file access utilities**

Name	Type	Description
fappend	verb	append text to file
fappends	verb	append string to file
fcopynew	verb	copies files if changed
fcopynews	verb	copies files as strings if changed
fdir	verb	file directory
ferase	verb	erases a file
fexist	verb	test if a file exists
fread	verb	read file
freadr	verb	read records from flat file
freads	verb	read file as string
freplace	verb	replace text in file
fselect	verb	file selection dialog
fsize	verb	return file size
fss	verb	file string search
fssrplc	verb	file string search and replace
fstamp	verb	returns file timestamp
fview	verb	view file
fwrite	verb	write text to file
fwrites	verb	write string to file

**fappend**

append text to file

The text is first ravelled. The file is created if necessary. Returns number of characters written, or an error message.

form: text fappend filename

example:

```
'chatham' fappend 'newfile.txt'
```

7

## **fappends**

append string to file

The text is first ravelled into a vector with each row terminated by the CRLF pair. Any single LF or CR characters in the text are converted into the CRLF pair. The file is created if necessary. Returns number of characters written, or an error message.

## **fcopynew**

copies files if changed

form: tofile fcopynew fromfiles

returns: 0, size      not changed  
          1, size      changed  
          \_1            failure

## **fcopynews**

copies files as strings if changed

form: tofile fcopynews fromfiles

returns: 0, size      not changed  
          1, size      changed  
          \_1            failure

## **fdir**

file directory

example:

```
fdir jsystempath 'system\main\s*.ijs'
```

## **ferase**

erases a file

Returns 1 if successful, otherwise \_1

## **fexist**

test if a file exists

Returns 1 if the file exists, otherwise 0.

## **fread**

read file

y. is filename {;start size}

x. is optional:

= b read as boxed vector

= m read as matrix

= s read as string (same as freads)

## **freadr**

read records from flat file

y. is filename {;record start, # of records}

records are assumed of fixed length delimited by one (only) of CR, LF, or CRLF.

the result is a matrix of records.

## **freads**

read file as string

y. is filename {;start size}

x. is optional (b and m same as fread):

= b read as boxed vector

= m read as matrix

freads

## **freplace**

replace text in file

form: dat freplace file;pos

## **fselect**

file selection dialog

y. = DOS filespec or ''

x. = optional file type list

returns user selection

## **fsize**

return file size

returns file size or \_1 if error

**fss**

file string search

form: str fss file

search file for string, returning indices

**fssrplc**

file string search and replace

form: (old;new) fssrplc file

**fview**

view file

uses standard Windows edit control,  
which is limited to around 20K size.

## format: formatting utilities

Name	Type	Description
center	verb	center text in given width
clipfmt	verb	format data for clipboard
clipunfmt	verb	unformat data read from clipboard
colhdr	verb	define column headers
expandby	conj	expand data with a given value
expandn	verb	expand data at every nth item
flatten	verb	flatten array to a character string
fmt	verb	format a numeric matrix, various format specs
fold	verb	fold text to width
hexdump	verb	show text as hex and ascii characters
nfmt	verb	simple numeric formatter
ruler	verb	formatted ruler
sqzint	verb	squeeze list of positive integers into short form
sqzrun	verb	squeeze list of numbers into short form
xfmt	verb	format extended integers

### center

center text in given width  
 form: width center text

### clipfmt

format data for clipboard  
 format array of rank 0 1 or 2 for clipboard.  
 columns are separated by TAB, rows by CRLF.

### clipunfmt



unformat data read from clipboard  
returns boxed matrix from clipboard result,  
recognizing TAB and CRLF as separators.  
characters are not converted to numbers.  
note this is not a true inverse of clipfmt.  
e.g. try: clipunfmt clipfmt i.5 6

## **colhdr**

define column headers  
returns matrix of column headers.

y. = list with columns delimited by semicolons; and lines  
in each column delimited by commas.

x. is wid or (wid;just), where:

wid = column widths

just = a singleton or vector of:

0 = centre header, then right justify (default)

1 = center header in wid

2 = right justify

3 = left justify

e.g. hdr=.

```
'Number,of,employees;Total,salary;Monthly,net,payment'  
(15 12 12;1) colhdr hdr
```

## **expandby**

expand data with a given value

e.g.     0 1 0 0 1 expandby 99 [ 10 20  
         99 10 99 99 20

## **expandn**

expand data at every nth item

n expandn dat

expand array at every nth cell.

e.g. 'ABC DEF G' = 3 expandn 'ABCDEFGG'

## **flatten**

flatten array to a character string

flattens array to a character string with same display

## **fmt**

format a numeric matrix, various format specs

syntax: specs fmt nums

nums = numeric vector or matrix or boxed list  
a vector is treated as a 1-row matrix.  
a boxed list is treated as boxed columns  
specs= formats, separated by commas, applied  
to each column, or item if boxed.

formats are either edit or positional:

edit formats have the form: {o}w{.d}, where:

o is optional qualifiers from the set:  
nr=n repetitions (must be given first)  
b=blank if zero  
c=commas  
z=zero fill  
w is field width  
d is decimal places

positional formats have the form:

xn = n blanks, e.g. x3

e.g. 'c10.3,x2,2rz5' has formats:

width 10, decimal places 3, commas  
2 blanks  
width 5, zero fill, repeated twice

## **fold**

fold text to width

syntax: {width} fold data

data is character vector

width defaults to screenwidth

## **nfmt**

simple numeric formatter

simple format of numeric vector or matrix in readable form.

opt is optional, up to 3 elements:

0 = maximum decimal places, max 9 (4)  
1 = minimum field width (0)  
2 = display width (screenwidth)

**ruler**

formatted ruler

returns a formatted ruler

e.g.      ruler 75          horizontal  
         1 ruler 30          vertical

**sqzint**

squeeze list of positive integers into short form

Squeeze list of positive integers into short character list.

If x. = 1, sort y. first.

e.g.   sqzint 1 2 3 7 8 9 10   =   1-3,7-10

see also <sqzrun>

**sqzrun**

squeeze list of numbers into short form

Squeeze list of numbers into short character list.

e.g.   sqzrun 1.1 1.1 1.1 7 9 9 10.25 = 3#1.1,7,2#9,10.25

See also <sqzint>.

**xfmt**

format extended integers

form: [width] xfmt number

groups in 3's up to 1e12, and 5's thereafter

**graph: graph package**

Name	Type	Description
gdarc	verb	draw arc
gdarc01	verb	draw arc in 0 0 1 1
gdchord	verb	draw chord
gdchord01	verb	draw chord in 0 0 1 1
gdcolor	verb	set color
gddraw	verb	wrapper for draw methods
gdellipse	verb	draw ellipse
gdellipse01	verb	draw ellipse in 0 0 1 1
gdlines	verb	draw lines
gdlines01	verb	draw lines in 0 0 1 1
gdopen	verb	open/clear graphics window
gdpen	verb	set pen size and style
gdpencolor	verb	set pen color
gdpie	verb	draw pie-shaped wedge
gdpie01	verb	draw pie-shaped wedge in 0 0 1 1
gdpixel	verb	draw pixel
gdpixel01	verb	draw pixel in 0 0 1 1
gdpolygon	verb	draw
gdpolygon01	verb	draw in 0 0 1 1
gdirect	verb	draw rectangle
gdirect01	verb	draw rectangle in 0 0 1 1
gdroundr	verb	rounded rectangle
gdroundr01	verb	rounded rectangle in 0 0 1 1
gdshow	verb	show graphics

**isigraph: supports graphics using the isigraph control**

Name	Type	Description
cile	verb	x. cile values of y.
fitrect01	verb	fit rectangle data to 0 1
fitrect11	verb	fit rectangle data to _1 1
fit01	verb	fit data to range 0 1
fit11	verb	fit data to range _1 1
gbitmap	verb	bitmap viewer using isigraph control
gbrush	verb	set brush color
gclear	verb	clear graphics window
gfit	verb	fit data to graphics window
glines	verb	display line connecting points
gopen	verb	open graphics window
gpen	verb	set pen color [;width,style]
gpolygon	verb	draw polygon given points
grayscale	verb	generate grayscale
grgb	verb	set color values
gscale	verb	scale from (-1,1) to (0,1000)
gshow	verb	{parent} gshow graph
hue	verb	generate color from color set
hueRGB	verb	generate color from RGB color se
polygon	verb	vertices of a regular polygon
rotate	verb	rotate angle by given amount

**cile**

x. cile values of y.

example:

3 cile i.12

### **fitrect01**

fit rectangle data to 0 1  
form: [anisotropic] fitrect01 data

if left argument is 1, applies same factors to both columns  
otherwise, scales columns independently.

### **fit01**

fit data to range 0 1  
form: [anisotropic] fit data

if left argument is 1, scales columns independently,  
otherwise applies same factors to all data,

### **gbitmap**

bitmap viewer using isigraph control  
y. is a boolean matrix

Note max size is about 80 x 80 - after which the graphics buffer  
is full.

This function is fast for small bitmaps. Use viewbmp in bmp.ijs  
to display larger bitmaps.

e.g. gshow gbitmap ?40 40\$2 [gopen''

### **gfit**

fit data to graphics window

scales each column of data independently to range (0,1000)

### **glines**

display line connecting points

{color;width;style} glines points

points should be a 2 column matrix of xy values

**gopen**

open graphics window  
y. = controlname;parentname  
    if either empty, default to 'isigraph'  
e.g.    gopen ''  
        gopen '';'J Graphics'

**gpen**

set pen color [;width,style]  
(default 1,0}

**gpolygon**

draw polygon given points  
{color} gpolygon points

**grayscale**

generate grayscale  
example:  
    grayscale (i.%<:) 5

**hue**

generate color from color set  
x. is color set  
y. is values from 0 to 1, selecting color

**hueRGB**

generate color from RGB color set  
x. is RGB color set (default HUES)  
y. is values from 0 to 1, selecting color

**polygon**

vertices of a regular polygon  
y. = number of points  
x. = scale factor on angle  $2\pi/y$ . (default 1)  
  
e.g.    polygon 5 = pentagon

2 polygon 5 = 5 pointed star

**rotate**

rotate angle by given amount

e.g. 0.5p1 rotate points = rotate clockwise a quarter turn



**jfiles: component file definitions**

Name	Type	Description
jappend	verb	append to jfile, (<i.2 3) jappend 'dat'
jcreate	verb	create jfile, jcreate 'dat'
jdup	verb	duplicate jfile, 'new' jdup 'dat'
jerase	verb	erase jfile, jerase 'dat'
jread	verb	read jfile, jread 'dat';2
jreplace	verb	replace in jfile, ('new value';123) jreplace 'dat';2 5
jsize	verb	return size of jfile, jsize 'dat'

**jmf: mapped files**

Name	Type	Description
additem	verb	add item to mapped noun
createjmf	verb	create mapped file
map	verb	map a file
memshare	verb	share memory with a process
memshareclose	verb	close memory shared with memshare
share	verb	share a mapped file
showmap	verb	show all mappings
unmap	verb	unmap a mapped file
unmapall	verb	unmap all mapped files

**createjmf**

create mapped file  
 createjmf fn;msize

**map**

map a file  
 [type [,trailing\_shape]] map name;filename [;sharename;ro]

**memshare**

share memory with a process

Form: {mapping-address} =: memshare {'share-name'}

this permits sharing memory with a non-J process

memshare and memshareclose contributed by Tony Zackin

**memshareclose**

close memory shared with memshare

Form: memshareclose {'share-name'}

### **unmap**

unmap a mapped file

[force] unmap name [;newsizel - 0 ok, 1 not mapped, 2 refs

### **unmapall**

unmap all mapped files

[force] unmapall dummy - unmap all

**jselect: provides jselect form**

Name	Type	Description
create	verb	create select form
destroy	verb	destroy select form

**keyfiles: keyed-file definitions**

Name	Type	Description
keycreate	verb	create file
keydir	verb	keyword directory
keydrop	verb	drop keywords
keyerase	verb	erase file
keyread	verb	read data
keyreadx	verb	read extra data
keywrite	verb	write data
keywritex	verb	write extra data
key_new	verb	make new components

**key\_new**

make new components

form: key\_new number, handle, used

**kfiles: older keyed-file definitions, replaced by keyfiles**

Name	Type	Description
kcreate	verb	create file
kdir	verb	keyword directory
kerase	verb	erase file
kread	verb	read data for keyword
kwrite	verb	write data for keyword

**menu: popup menu**

Name	Type	Description
wdmenu	verb	provide a pop-up menu

## misc: miscellaneous utilities

Name	Type	Description
addwmfheader	verb	add metafile header to wmf file
boxcols	verb	box columns of matrix
chop	verb	chop array into boxed list
default	verb	set default value
diff	verb	second differences
index	verb	index where result is _1 if not found, instead of #x.
join	verb	join boxed items
nubcount	verb	nub + count
pathname	verb	split DOS name into path;name
prompt	verb	prompt for input
scriptform	verb	script representation of names
show	verb	show names using linear representation
subs	conj	substitution
tolist	verb	convert boxed list to LF delimited list

### addwmfheader

add metafile header to wmf file

form: [outfile] addwmfheader infile

metafiles used by Word etc. require 22 byte header

outfile addwmfheader infile ; width height (%1000 of inches)

### boxcols

box columns of matrix

y. is a matrix

x. indicates partitions

- a single integer is size of each partition



- a boolean is beginning of each partition  
examples:

```
3 boxcols i.3 7
+-----+-----+---+
| 0  1  2| 3  4  5| 6|
| 7  8  9|10 11 12|13|
|14 15 16|17 18 19|20|
+-----+-----+---+
```

```
1 0 1 0 0 0 1 boxcols i.3 7
+-----+-----+---+
| 0  1| 2  3  4  5| 6|
| 7  8| 9 10 11 12|13|
|14 15|16 17 18 19|20|
+-----+-----+---+
```

## chop

chop array into boxed list

chop character vector or matrix into boxed list.

x. is optional delimiter, default LF if in text, else blank.

If a matrix, the delimiter must be vertically aligned,  
otherwise use chop"1 to chop each row of the matrix.

e.g. chop ": 10 20 30

chop ": i. 5 4

## default

set default value

name default value

set global name to value if not already defined

## index

index where result is \_1 if not found, instead of #x.

example:

```
'abc' index 'ce'
2 _1
```

## prompt

prompt for input

prompts for input, optionally with a default result

form: [default] prompt prompt\_text

examples:

```
prompt 'start date: '  
'2001 5 23' prompt 'start date: '
```

## **scriptform**

script representation of names

representation using multi-line script form for most explicit definitions, otherwise linear representation.

useful for writing object definitions to a script file.

## **show**

show names using linear representation

show names using linear representation to screen width

syntax:

```
show namelist    (e.g. show 'deb edit list')  
show numbers     (from 0 1 2 3=nouns, adverbs etc)  
show ''          (equivalent to show 0 1 2 3)
```

useful for a quick summary of object definitions

## **subs**

substitution

form: new (old subs test) data

examples:

```
10 (2 subs =) 1 3 2 1 5 2  
1 3 10 1 5 10  
10 (2 subs <:) 1 3 2 1 5 2  
1 10 10 1 10 10  
(from David Alis)
```

**myutil: example scripts for user definitions**

Name	Type	Description
brep	verb	boxed representation
lrep	verb	linear representation
pps	verb	set print precision
prep	verb	parenthesized representation
snap	verb	names snapshot
time	verb	time
timespacex	verb	time and space for expressions
timex	verb	time expressions
tolist	verb	convert boxed list to LF delimited list
tree	verb	tree representation

**timespacex**

time and space for expressions

Form: [repetitions] timex 'expression'

Example:

```
10 timespacex &> 'q:123456787';'3^10000x'
0.005 58432
0.061 52352
```

**timex**

time expressions

Form: [repetitions] timex 'expression'

**nfiles: read/write native files in various formats**

Name	Type	Description
nappend	verb	append to file
nread	verb	read file
nwrite	verb	write to file

**nappend**

append to file

left argument is data

**nwrite**

write to file

left argument is data

**numeric: numeric utilities**

Name	Type	Description
baserep	verb	y. in base x.
clean	verb	clean y. to tolerance of x. (default 1e_10)
colsum	verb	sum data columns of matrix by key
groupndx	verb	group indices of y. in x.
int01	verb	interval in n steps from 0 to 1 (= steps 0 1,n)
linsert	verb	linear insert x. (default 2) steps into y.
randomize	verb	sets a random value into random link
range	verb	range from a to b [in steps of c]
recur	verb	solves recurrence $r(i)=a(i)+r(i-1)*m(i-1)$
round	verb	round y. to nearest x. (e.g. 1000 round 12345)
rounddist	verb	round y. to nearest x. preserving total
roundint	verb	round to nearest integer
steps	verb	steps from a to b in c steps

**clean**

clean y. to tolerance of x. (default 1e\_10)  
 form: tolerance (default 1e\_10) clean numbers  
 sets values less than tolerance to 0

**colsum**

sum data columns of matrix by key  
 form: key colsum mat  
 sum data columns of matrix on key columns  
 e.g. if column 2 of mat is age, then  
     2 colsum mat  
 sums the remaining columns by age

### **groupndx**

group indices of y. in x.

Return group indices of elements of y.

x. is an integer vector of the starting numbers of each group,  
assumed to be in ascending order.

e.g. 0 0 0 1 1 1 2 2 = 0 3 6 groupndx i.8

i.e. <:@(+/@(<:/))

### **recur**

solves recurrence  $r(i)=a(i)+r(i-1)*m(i-1)$

form: r = m recur a

r(0) = a(0)

r(i) = a(i)+r(i-1)\*m(i-1)

e.g. 1.05 1.10 recur 100 100 100

100 205 325.5

### **rounddist**

round y. to nearest x. preserving total

distributive rounding

round y. to nearest x. preserving total to nearest x.

e.g. 0.1 rounddist 6\$0.45

0.5 0.5 0.5 0.4 0.4 0.4

### **steps**

steps from a to b in c steps

form: steps a,b,c

**pack: package utilities**

Name	Type	Description
pack	verb	package namelist
packlocale	verb	package locales
pcompare	verb	compare two packages
pdef	verb	package define
pex	verb	remove namelist from package
pget	verb	return value of name in package
psel	verb	select namelist from package
pset	verb	merge new into old

**pack**

package namelist

```
form:  pack 'one two three'
       pack 'one';'two';'three'
```

**packlocale**

package locales

```
form: packlocale locales
```

```
example: packlocale 'base';'z';'j'
```

each locale is packaged and forms one row of the result

**pcompare**

compare two packages

```
form: pk1 pcompare pk2
```

**pdef**

package define

form: pdef pk

### **pex**

remove namelist from package

form: namelist pex pk

### **pget**

return value of name in package

form: name pget pk                      - return value of name in package

### **psel**

select namelist from package

form: namelist psel pk

### **pset**

merge new into old

form: new pset old

result has values in new, plus any values in old that  
were not replaced in new



**parts: partition functions**

Name	Type	Description
firstones	verb	first 1's in partition
lastones	verb	last 1's in partition
lfp	verb	lengths from partition
partition	verb	partition items (1 marks new item)
pfl	verb	partition from lengths
preverse	verb	partitioned reverse
psort	verb	partitioned sort
psum	verb	partitioned sum
psumscan	verb	partitioned sumscan
runindices	verb	indices from run lengths
runlengths	verb	lengths of each run

**firstones**

first 1's in partition

form: firstones part

**lastones**

last 1's in partition

form: lastones part

**lfp**

lengths from partition

form: lfp part

**partition**

partition items (1 marks new item)

form: partition dat

partition=: 1: , }. ~: }:

### **pfl**

partition from lengths  
form: pfl len

### **preverse**

partitioned reverse  
form: part preverse dat

### **psort**

partitioned sort  
form: part psort dat

### **psum**

partitioned sum  
form: part psum dat

### **psumscan**

partitioned sumscan  
form: part psumscan dat

### **runindices**

indices from run lengths  
form: runindices

### **runlengths**

lengths of each run  
form: runlengths dat

**plot: session definitions for plot**

Name	Type	Description
pd	verb	plot driver
plot	verb	standard plot

**plot**

standard plot

in jwplot locale:

PForm      form name

pclose     close form

**print: print package**

Name	Type	Description
print	verb	print text
printfile	verb	print file
printfile2	verb	print file in 2-up mode
print2	verb	print text in 2-up mode
xtab	verb	remove tab stops from character string
xtabline	verb	remove tab stops

**xtabline**

remove tab stops

remove tabs from single line

**publish: publish J code to HTML**

Name	Type	Description
fmtlines	verb	format lines of J code
fmtscript	verb	format script
fmtscriptto	verb	format script
publish	verb	interactive publisher

**fmtlines**

format lines of J code  
 fmtlines 'J code'

**fmtscript**

format script  
 fmtscript 'script.ijs' NB. saved as script.htm

**fmtscriptto**

format script  
 'script.ijs' fmtscriptto 'output.htm' B. defaults (note default  
 for BROWSER done at runtime)

**publish**

interactive publisher  
 publish '' or publish 'scriptname'

**random: various random number utilities**

Name	Type	Description
deal	verb	deal x. items from y. (no repetition)
dealx	verb	deal x. indices from list y. (no repetition)
randomize	verb	sets a random value into random link
rand01	verb	generate y. random numbers in interval (0,1)
rand11	verb	generate y. random numbers in interval (_1,1)
setrl	verb	set random link
toss	verb	pick x. items from y. (with repetition)
tossx	verb	pick x. indices from list y. (with repetition)

**tossx**

pick x. indices from list y. (with repetition)

-----  
examples:

```

p=: ;: 'anne dave mary tom'
words=: ;: inverse

words deal p
mary anne dave tom

words 3 deal p
tom dave mary

words 6 toss p
mary mary dave dave mary tom

dealx 2 3
0 1
1 1
0 2
1 2
0 0

```

1 0

3 dealx 3 5 7

2 4 6

1 0 6

2 4 5

4 tossx 2 3

0 2

1 2

1 1

0 2

**regex: Regular expression pattern matching**

Name	Type	Description
rxall	verb	regex equivalent of { (all matches)
rxapply	verb	apply verb to pattern
rxcomp	verb	compile pattern
rxcut	verb	cut string into nomatch/match list
rxeq	verb	regex equivalent of -:
rxerror	verb	last regex error message
rxE	verb	regex equivalent of E.
rxfirst	verb	regex equivalent of {.@{ (first match)
rxfree	verb	free pattern handles
rxfrom	verb	matches from string
rxhandles	verb	list pattern handles
rxin	verb	regex equivalent of e.
rxindex	verb	regex equivalent of i.
rxinfo	verb	info on pattern handles
rxmatch	verb	single match
rxmatches	verb	all matches
rxmerge	verb	replace matches in string
rxrplc	verb	search and replace



**rgb: convert between color triples and RGB values**

Name	Type	Description
RGB	verb	convert between color triples and RGB values

**socket: winsock package**

Name	Type	Description
sdaccept	verb	accept connection
sdasync	verb	set up async connection for the socket
sdbind	verb	bind socket
sdcheck	verb	check socket for errors
sdcleanup	verb	initialize winsock
sdclose	verb	close socket
sdconnect	verb	connect to the socket
sdgethostbyaddr	verb	returns a name from an address
sdgethostbyname	verb	returns an address from a name
sdgethostname	verb	returns host name
sdgetpeername	verb	returns address this socket is connected to
sdgetsockets	verb	returns all socket numbers
sdgetsockname	verb	returns address of this socket
sdinit	verb	initialize winsock
sdioctl	verb	read or write socket control information
sdionread	verb	get number of bytes available for reading socket
sdlisten	verb	set up listener for the socket
sdrecv	verb	read data
sdrecvfrom	verb	read data from
sdselect	verb	find sockets that need work
sdsend	verb	send data
sdsendto	verb	send data
sdsetsockopt	verb	sets the value of a socket option.
sdsockaddress	verb	returns address

sdsockerror	verb	retrieve socket error code
sdssocket	verb	creates a socket

### **sdaccept**

accept connection

y. - socket

### **sdbind**

bind socket

y. - socket , family , address , port

### **sdcheck**

check socket for errors

socket utilities

### **sdclose**

close socket

y. - socket

### **sdconnect**

connect to the socket

y. - socket , family , address , port

### **sdgethostbyaddr**

returns a name from an address

y. - AF\_INET;host ip address

### **sdgethostbyname**

returns an address from a name

y. - host name

### **sdgetpeername**

returns address this socket is connected to

y. active socket

**sdgetsockname**

returns address of this socket  
y. active socket

**sdlisten**

set up listener for the socket  
y. - socket;queue\_length  
SOMAXCONN - The maximum length of the queue of pending connections

**sdrecv**

read data  
y.- socket,data\_size,An indicator specifying the way in which the call is made (0)

**sdrecvfrom**

read data from  
y.- socket,data\_size,An indicator specifying the way in which the call is made (0)

**sdsend**

send data  
y.- socket;An indicator specifying the way in which the call is made (0)  
x.- data

**sdsendto**

send data  
y.- socket ; flags ; family ; address ; port  
x.- data

**sdsockaddress**

returns address  
y. active socket

**statdist: statistical distributions**

Name	Type	Description
betarand	verb	random numbers in a beta distribution
binomialdist	verb	discrete values for binomial distribution
binomialprob	verb	probability of success in binomial distribution
binomialrand	verb	random numbers 0 and 1 in binomial distribution
cauchyrand	verb	random numbers in a cauchy distribution
discreterand	verb	random numbers in a discrete distribution
exponentialrand	verb	random numbers in an exponential distribution
gammarand	verb	random numbers in a gamma distribution
normalprob	verb	probability of success in normal distribution
normalrand	verb	random numbers in a standard normal distribution,
poissondist	verb	discrete values for poisson distribution
poissonprob	verb	probability of success in poisson distribution
poissonrand	verb	random numbers in a poisson distribution

**statfns: statistical functions**

Name	Type	Description
cile	verb	x. cile values of y.
comb	verb	combinations of size x. from i.y
corr	verb	correlation
cov	verb	covariance
dev	verb	deviation from mean
dstat	verb	descriptive statistics
freqcount	verb	frequency count
geomean	verb	geometric mean
harmean	verb	harmonic mean
kurtosis	verb	kurtosis
lsfit	verb	least-squares fit
max	verb	maximum
mean	verb	arithmetic mean
median	verb	median
midpt	verb	index of midpoint
min	verb	minimum
perm	verb	permutations of size y.
regression	verb	multiple regression
skewness	verb	skewness
spdev	verb	sum of products of deviations
ssdev	verb	sum squares of deviation
stddev	verb	standard deviation
var	verb	variance

**stdlib: standard library**

Name	Type	Description
adverb	noun	integer 1
assert	verb	assert value is true
bind	conj	binds argument to a monadic verb
boxopen	verb	box argument if open
boxxopen	verb	box argument if open and 0<#
bx	verb	indices of 1's in boolean
clear	verb	clear all names in locale
conjunction	noun	integer 2
cutopen	verb	cut argument if open
CR	noun	carriage return character
CRLF	noun	CR LF pair
datatype	verb	noun datatype
def	conj	: (explicit definition)
define	adverb	: 0 (explicit definition script form)
do	verb	name for ".
drop	verb	name for }.
dyad	noun	integer 4
each	adverb	each (&.>)
empty	verb	return empty result (i.0 0)
erase	verb	erase namelist
every	adverb	every (&>)
expand	verb	boolean expand
EAV	noun	ascii 255 character
fetch	verb	name for {::

FF	noun	formfeed character
inverse	adverb	inverse (^:_1)
leaf	adverb	leaf (L:0)
list	verb	list data formatted in columns
LF	noun	linefeed character
monad	noun	integer 3
nameclass	verb	name for 4!:0
namelist	verb	name for 4!:1
names	verb	formatted namelist
nc	verb	name for 4!:0
nl	verb	selective namelist
noun	noun	integer 0
on	conj	name for @:
pick	verb	pick (>@ {)
rows	adverb	rows ("1)
script	verb	load script, cover for 0!:0
scriptd	verb	load script with display, cover for 0!:1
sign	adverb	sign (*)
smoutput	verb	output to session
sort	verb	sort up
split	verb	split head from tail
table	adverb	function table
take	verb	name for {.
toCRLF	verb	converts character strings to CRLF delimiter
toHOST	verb	converts character strings to Host delimiter
toJ	verb	converts character strings to J delimiter (linefeed)
tolower	verb	convert text to lower case
toupper	verb	convert text to upper case



type	verb	object type
TAB	noun	tab character
verb	noun	integer 3
wcsize	verb	size of execution window

## **assert**

assert value is true  
 assertion failure if 0 e. y.  
 e.g. 'invalid age' assert 0 <: age

## **bind**

binds argument to a monadic verb  
 binds monadic verb to an argument creating a new verb  
 that ignores its argument.  
 e.g. fini=: wdinfo bind 'finished...'

## **boxopen**

box argument if open  
 boxopen - box argument if open and # is not zero  
 e.g. if script=: 0!:(0 @ boxopen), then either  
     script 'work.ijs' or script <'work.ijs'  
 use cutopen to allow multiple arguments.

## **clear**

clear all names in locale  
     returns any names not erased  
 example: clear 'myloc'

## **cutopen**

cut argument if open  
 this allows an open argument to be given where a boxed list is  
 required.  
 most common situations are handled. it is similar to boxopen,  
 except  
 allowing multiple arguments in the character string.

x. is optional delimiters, default LF if in y., else blank

y. is boxed or an open character array.

if y. is boxed it is returned unchanged, otherwise:  
if y. has rank 2 or more, the boxed major cells are returned  
if y. has rank 0 or 1, it is cut on delimiters in given in x., or  
if x. not given, LF if in y. else blank. Empty items are  
deleted.

e.g. if script=: 0!:0 @ cutopen, then  
script 'work.ijs util.ijs'

## **expand**

boolean expand  
form: boolean expand data

## **list**

list data formatted in columns  
syntax: {width} list data  
accepts data as one of:  
boxed list  
character vector, delimited by CR, LF or CRLF; or by ' '  
character matrix  
formats in given width, default screenwidth

## **nl**

selective namelist  
Form: [mp] nl sel

sel: one or more integer name classes, or a name list.  
if empty use: 0 1 2 3.  
mp: optional matching pattern. If mp contains '\*', list names  
containing mp, otherwise list names starting mp. If mp  
contains '~', list names that do not match.

e.g. 'f' nl 3 - list verbs that begin with 'f'  
'\*com nl '' - list names containing 'com'

## **split**

split head from tail  
examples:

```
split 'abcde'  
2 split 'abcde'
```

## **table**

```
function table  
table    - function table (adverb)  
e.g.    1 2 3 * table 10 11 12 13  
        +. table i.13
```

## strings: string manipulation

Name	Type	Description
charsub	verb	character substitution
cut	verb	cut text, by default on blanks
cuts	verb	cut y. at x. (conjunction)
deb	verb	delete extra blanks
delstring	verb	delete occurrences of x. from y.
dlb	verb	delete leading blanks
dltb	verb	delete leading and trailing blanks
dltps	verb	delete multiple leading and trailing blanks
dropafter	verb	drop after x. in y.
dropto	verb	drop to x. in y.
dtb	verb	delete trailing blanks
dtbs	verb	delete multiple trailing blanks in text
fstringreplace	verb	file string replace
ljust	verb	left justify
rjust	verb	right justify
rplc	verb	replace characters in text string
ss	verb	string search
stringreplace	verb	replace characters in text string
takeafter	verb	take after x. in y.
taketo	verb	take to x. in y.

### charsub

character substitution  
 characterpairs charsub data  
 For example:

```
'_-$ ' charsub '$123 -456 -789'
123 _456 _789
Use <rplc> for arbitrary string replacement.
```

### **cuts**

```
cut y. at x. (conjunction)
string (verb cuts n) text
  n=_1 up to but not including string
  n= 1 up to and including string
  n=_2 after but not including string
  n= 2 after and including string
```

### **dltbs**

```
delete multiple leading and trailing blanks
text is delimited by characters in x. with default LF
example:
```

```
< 'A' dltbs ' A abc  def  Ars  A  x y  z  '
+-----+
|Aabc  defArsAx y  z|
+-----+
```

### **dtbs**

```
delete multiple trailing blanks in text
text is delimited by characters in x. with default CRLF
example:
```

```
< 'A' dtbs ' A abc  def  Ars  A  x y  z  '
+-----+
|A abc  defArsA  x y  z|
+-----+
```

Algorithm thanks to Brian Bambrough (JForum Nov 2000)

### **fstringreplace**

```
file string replace
form: (old;new) fstringreplace file
```

### **rplc**

```
replace characters in text string
```

form: text rplc oldnew  
oldnew is a 2-column boxed matrix of old ,. new  
or a vector of same

replace priority is the same order as oldnew

Examples:

```
'ababa' rplc 'aba';'XYZT';'ba';'+'  
XYZT+
```

```
'ababa' rplc 'ba';'+';'aba';'XYZT'  
a++
```

## **stringreplace**

replace characters in text string

form: oldnew stringreplace tet  
oldnew is a 2-column boxed matrix of old ,. new  
or a vector of same

stringreplace priority is the same order as oldnew

Examples:

```
('aba';'XYZT';'ba';'+') stringreplace 'ababa'  
XYZT+
```

```
('ba';'+';'aba';'XYZT') stringreplace 'ababa'  
a++
```

**sysenv: sysenv - System Environment**

Name	Type	Description
ADDON	noun	path to addons
ARGV	noun	command line
CONFIG	noun	path to user configuration files
FIXFONT	noun	fixed space font (used in session windows)
IFCONSOLE	noun	if a console front end
IFJAVA	noun	if a Java front end
IFUNIX	noun	if UNIX
IFWINCE	noun	if Windows CE
IFWINNT	noun	if Windows NT,2000,XP
IFWIN32	noun	if Windows (9x,ME,NT,2000,XP)
jaddonpath	verb	adds path to addons
jconfigpath	verb	adds path to user configuration files
jhostpath	verb	converts path name to use host path separator
jsystemdefs	verb	loads appropriate netdefs or hostdefs
jsystempath	verb	adds path to J system directory
jtemppath	verb	adds path to directory for temporary files
juserpath	verb	adds path to user's directory
JIJX	noun	1 if -jijx parameter (i.e. don't create ijx window)
PROFILE	noun	name of profile file
PROFONT	noun	proportional font (used in forms)
SYSTEM	noun	path to J system directory
TEMP	noun	path to temporary directory for temporary files
UNAME	noun	name of UNIX o/s
USER	noun	path to user's directory

**text: text utilities**

Name	Type	Description
capitalize	verb	capitalize text
cutpara	verb	cut text into boxed list of paragraphs
cuttext	verb	cut text into boxed list of texts
foldpara	verb	fold single paragraph
foldtext	verb	fold text to given width
topara	verb	convert text to paragraphs

**capitalize**

capitalize text

capitalize text (vector delimited by LF, or matrix)

all first letters are capitalized, otherwise:

x.=0 capitalize first letter following a fullstop followed by  
a blank or LF or LF,LF (sentence capitalization=default)

1 capitalize any letter preceded by a blank

2 capitalize first letter in any alphabetic string

**cutpara**

cut text into boxed list of paragraphs

form: cutpara text

**cuttext**

cut text into boxed list of texts

form: cuttext text

**foldpara**

fold single paragraph

syntax: {width} fold data

data is character vector



**foldtext**

fold text to given width  
form: width foldtext text

**topara**

convert text to paragraphs  
form: topara text  
replaces single LF's not followed by blanks by spaces,  
except for LF's at the beginning

**trig: trigonometric functions**

Name	Type	Description
arccos	verb	arccos
arccosh	verb	arccosh
arcsin	verb	arcsin
arcsinh	verb	arcsinh
arctan	verb	arctan
arctanh	verb	arctanh
cos	verb	cos
cosd	verb	cos in degrees
cosh	verb	cosh
dfr	verb	degrees from radians
indegrees	adverb	convert function to use degrees:
pi	noun	pi
rfd	verb	radians from degrees
sin	verb	sin
sind	verb	sin in degrees
sinh	verb	sinh
tan	verb	tan
tand	verb	tan in degrees
tanh	verb	tanh

**validate: data validation**

Name	Type	Description
inrange	verb	(low, high) inrange data
isbalanced	verb	pair isbalanced string
isboolean	verb	data is all 0 or 1
isboxed	verb	is boxed
ischaracter	verb	data is character
iscomplex	verb	data is complex
iscounter	verb	data is non-negative integer (counting number)
isdate	verb	is date (as yyyy mm dd)
isinteger	verb	data is all integer
ismatrix	verb	data is a matrix
isnumeric	verb	data is numeric
isreal	verb	data is all real
isscalar	verb	data is a scalar
isunicode	verb	data is unicode
isunique	verb	data has no duplicates
isvector	verb	data is a vector

**viewmat: viewmat init**

Name	Type	Description
gethue	verb	generate color from color set
viewmat	verb	view matrix in isigraph control

**gethue**

generate color from color set

x. is color set

y. is values from 0 to 1, selecting color

**viewmat**

view matrix in isigraph control

y. = [hue] mat [;title]] (as for viewbmp)

mat may be one of:

boolean	(black/white)
other numeric	(color scale from 0 upwards)
other	(converted to numeric)

hue is: 3 column table of R-G-B triples

or: list of RGB values

if mat is boolean, x. defaults to black/white

otherwise hue defaults to red - purple spectrum

e.g. viewmat (?50 50\$2);'';'Random Boolean'

## winapi: windows api utilities

Name	Type	Description
winconst	verb	look up Windows constants
winset	verb	set values of windows constants
win32api	adverb	look up Win32 API declaration, returning verb
win32apir	adverb	win32api, except verb returns first element of call result

### winconst

look up Windows constants

returns 2-element list: values ; names

e.g. winconst 'EM\_GETMODIFY EM\_SETMODIFY'

### win32api

look up Win32 API declaration, returning verb

e.g.

```
'GetProfileStringA' win32api
'kernel32 GetProfileStringA i *c *c *c *c i'&(15!:0)
  >'GetVersion' win32api ''
_1073741820
```

**winlib: standard windows library**

Name	Type	Description
wd	verb	main window driver, name for 11!:0
wdbox	verb	box wd argument
wdcenter	verb	center form on another
wdclipread	verb	read clipboard
wdclipwrite	verb	write to clipboard
wde	verb	as wd but displays error and signals break
wdfit	verb	fit form in window
wdget	verb	get values from matrix, e.g. wd'q'
wdhandler	verb	wd handler
wdinfo	verb	information box
wdisparent	verb	return 1 if a parent window
wdmove	verb	position window, relative to side of screen
wdpclose	verb	close parent window
wdqshow	verb	display result of wdq
wdquery	verb	query box
wdreset	verb	reset window driver
wdselect	verb	selection box
wdstatus	verb	put status message on screen
wdview	verb	text viewer

**wdbox**

box wd argument

use this to analyze arguments to wd

in code: whs=whitepace, del=delimiters

## **wdcenter**

center form on another

form: wdcenter xywh

use this to center a form on another

## **wdfit**

fit form in window

y. is two integers for horizontal and vertical

in each case, the entire form will be shown

values are:

0 move the side out of view back into the window

- typically reduces the form size

1 move the form so it is all visible

- typically leaves the form size unchanged

2 stretch the form to the window

3 maximize the form to full screen, hiding caption and borders

an empty argument is treated as 1 1

wd'qm' - return system metrics:

0-1 screen width, screen height,

2-3 x logical unit, y logical unit,

4-5 cxborder, cyborder,

6-7 cxfixedframe, cyfixedframe,

8-9 cxframe, cyframe,

10-11 cycaption, cymenu,

12-15 desktop xywh

## **wdget**

get values from matrix, e.g. wd'q'

utility to index 2-column boxed array, e.g. result of wd 'q'

form: names wdget data

returns items in second column indexed on names in first column

result is boxed if left argument is boxed

e.g. 'sysfocus' wdget wdq

## **wdhandler**

wd handler

runs in form locale

sets global event data: wdq

runs first found of: form\_handler, form\_event, form\_default,  
with global event variables from wdq  
if debug is off, wraps event handler in try. catch.  
catch exits if error message is the last picked up by debug.

## **wdinfo**

information box

syntax: wdinfo [title;] message

## **wdmove**

position window, relative to side of screen

form: [window] wdmove offset

offset is the xy offset.

if >: 0 the offset is from topleft

if < 0 the offset is from bottomright (less 1)

e.g.

0 0 = topleft

\_1 \_1 = bottomright

5 \_11 = 5 from left, 10 from bottom

## **wdqshow**

display result of wdq

display wdq result - useful for testing forms

## **wdquery**

query box

form: [opt] wdquery [title;] message

opt has one or two elements:

0{	= 0	okcancel	(ok=0 cancel=1)
	1	retrycancel	(retry=0 cancel=1)
	2	yesno	(yes=0 no=1)
	3	yesnocancel	(yes=0 no=1 cancel=2)
	4	abortretryignore	(abort=0 retry=1 ignore=2)
1{	= default button (0, 1 or 2)		

## **wdselect**

selection box



windows selection box

y. is a either:   boxed list of choices  
                  or:   title ; <boxed list of choices  
          if y. is empty, closes selection box if open.

x. is optional of up to 3 values (default 0). the second and  
  third options are only referenced when the box is created:  
  0{ initial selection  
  1{ 0=single selection, 1=multiple selection  
  2{ 0=close on exit, 1=leave open if selection made

returns 2 item boxed list:  
  0{ 0=cancel, 1=accept  
  1{ indices of selections

## **wdstatus**

put status message on screen  
write status message on screen

{title} wdstatus message       - write message  
      wdstatus ' '            - close message box

default text size is 1 row of 50 characters.  
for a larger size, call wdstatus initially with a message  
of the required size (pad with blanks if necessary).  
once created, the message box is not resized.

## **wdview**

text viewer  
y. is text or header;text [;wrap;print;top (default 0 1 0)]  
x. is optional window name  
uses standard Windows edit control,  
which is limited to around 20K in size.

**write: write package**

Name	Type	Description
drawbox	verb	draw box (outline or solid)
drawgraph	verb	draw graph (from wmf file)
drawline	verb	draw line
endpage	verb	end page
endprint	verb	end print job
makeframe	verb	make a frame
preview	verb	preview page
startpage	verb	start page
startprint	verb	initialize print job
write	verb	write text (see syntax below)

## Definition Summaries

Name	Source	Type	Description
additem	<a href="#">jmf p50</a>	v	add item to mapped noun
addwmfheader	<a href="#">misc p55</a>	v	add metafile header to wmf file
adverb	<a href="#">stdlib p70</a>	n	integer 1
arccos	<a href="#">trig p74</a>	v	arccos
arccosh	<a href="#">trig p74</a>	v	arccosh
arcsin	<a href="#">trig p74</a>	v	arcsin
arcsinh	<a href="#">trig p74</a>	v	arcsinh
arctan	<a href="#">trig p74</a>	v	arctan
arctanh	<a href="#">trig p74</a>	v	arctanh
assert	<a href="#">stdlib p70</a>	v	assert value is true
av	<a href="#">convert p37</a>	v	convert between characters and indices
ADDON	<a href="#">sysenv p72</a>	n	path to addons
AND	<a href="#">dll p44</a>	v	bitwise AND (&)
ARGV	<a href="#">sysenv p72</a>	n	command line
baserep	<a href="#">numeric p58</a>	v	y. in base x.
betarand	<a href="#">statdist p68</a>	v	random numbers in a beta distribution
bind	<a href="#">stdlib p70</a>	c	binds argument to a monadic verb
binomialdist	<a href="#">statdist p68</a>	v	discrete values for binomial distribution
binomialprob	<a href="#">statdist p68</a>	v	probability of success in binomial distribution
binomialrand	<a href="#">statdist p68</a>	v	random numbers 0 and 1 in binomial distribution
bitwise	<a href="#">dll p44</a>	a	bitwise operations

boxcols	<a href="#">misc p55</a>	v	box columns of matrix
boxopen	<a href="#">stdlib p70</a>	v	box argument if open
boxxopen	<a href="#">stdlib p70</a>	v	box argument if open and 0<#
brep	<a href="#">myutil p56</a>	v	boxed representation
bx	<a href="#">stdlib p70</a>	v	indices of 1's in boolean
calendar	<a href="#">dates p40</a>	v	calendar for year [months]
capitalize	<a href="#">text p73</a>	v	capitalize text
cauchyrand	<a href="#">statdist p68</a>	v	random numbers in a cauchy distribution
cd	<a href="#">dll p44</a>	v	call DLL procedure
cdcb	<a href="#">dll p44</a>	v	callback address
cder	<a href="#">dll p44</a>	v	error information
cderx	<a href="#">dll p44</a>	v	GetLastError information
cdf	<a href="#">dll p44</a>	v	free DLLs
center	<a href="#">format p46</a>	v	center text in given width
charsub	<a href="#">strings p71</a>	v	character substitution
chop	<a href="#">misc p55</a>	v	chop array into boxed list
cile	<a href="#">isigraph p48</a>	v	x. cile values of y.
cile	<a href="#">statfns p69</a>	v	x. cile values of y.
clean	<a href="#">numeric p58</a>	v	clean y. to tolerance of x. (default 1e_10)
clear	<a href="#">stdlib p70</a>	v	clear all names in locale
clipfmt	<a href="#">format p46</a>	v	format data for clipboard
clipunfmt	<a href="#">format p46</a>	v	unformat data read from clipboard
coclass	<a href="#">colib p33</a>	v	set current co class
cocreate	<a href="#">colib p33</a>	v	create object
cocurrent	<a href="#">colib p33</a>	v	set current locale
codestry	<a href="#">colib p33</a>	v	destroy current object

coerase	<a href="#">colib p33</a>	v	erase object
coextend	<a href="#">colib p33</a>	v	add class locale to path (before z)
cofind	<a href="#">coutil p38</a>	v	find objects containing name in object:
cofullname	<a href="#">colib p33</a>	v	return name with locale qualifier
coinfo	<a href="#">coutil p38</a>	v	return info on co classes
coinsert	<a href="#">colib p33</a>	v	insert into path (before z)
colhdr	<a href="#">format p46</a>	v	define column headers
colsum	<a href="#">numeric p58</a>	v	sum data columns of matrix by key
comb	<a href="#">statfns p69</a>	v	combinations of size x. from i.y
compare	<a href="#">compare p36</a>	v	compare character data
coname	<a href="#">colib p33</a>	v	return current co name
conames	<a href="#">colib p33</a>	v	formatted co name list
conew	<a href="#">colib p33</a>	v	create object
conjunction	<a href="#">stdlib p70</a>	n	integer 2
conl	<a href="#">colib p33</a>	v	return co name list
conouns	<a href="#">coutil p38</a>	v	nouns referencing object
conounsx	<a href="#">coutil p38</a>	v	object references with locales
copath	<a href="#">colib p33</a>	v	set/get co path
copathnl	<a href="#">coutil p38</a>	v	path name list
copathnlx	<a href="#">coutil p38</a>	v	formatted path name list with defining classes
coreset	<a href="#">colib p33</a>	v	destroy object locales,
corr	<a href="#">statfns p69</a>	v	correlation
cos	<a href="#">trig p74</a>	v	cos
cosd	<a href="#">trig p74</a>	v	cos in degrees
coselect	<a href="#">coutil p38</a>	v	select current locale

cosh	<a href="#">trig p74</a>	v	cosh
costate	<a href="#">coutil p38</a>	v	state of class objects
cov	<a href="#">statfns p69</a>	v	covariance
create	<a href="#">jselect p51</a>	v	create select form
createjmf	<a href="#">jmf p50</a>	v	create mapped file
cut	<a href="#">strings p71</a>	v	cut text, by default on blanks
cutopen	<a href="#">stdlib p70</a>	v	cut argument if open
cutpara	<a href="#">text p73</a>	v	cut text into boxed list of paragraphs
cuts	<a href="#">strings p71</a>	v	cut y. at x. (conjunction)
cuttext	<a href="#">text p73</a>	v	cut text into boxed list of texts
COLORTABLE	<a href="#">colortab p35</a>	n	main colortable
COLOR16	<a href="#">color16 p34</a>	n	table of HTML primary colors
CONFIG	<a href="#">sysenv p72</a>	n	path to user configuration files
CR	<a href="#">stdlib p70</a>	n	carriage return character
CRLF	<a href="#">stdlib p70</a>	n	CR LF pair
datatype	<a href="#">stdlib p70</a>	v	noun datatype
dberm	<a href="#">debug p42</a>	v	last error message
dberr	<a href="#">debug p42</a>	v	last error number
dbg	<a href="#">debug p42</a>	v	turn debugging window on/off
dbjmp	<a href="#">debug p42</a>	v	jump to line number
dblocals	<a href="#">debug p42</a>	v	display names and locals on stack
dblqx	<a href="#">debug p42</a>	v	latent expression query
dblxs	<a href="#">debug p42</a>	v	latent expression set
dbnxt	<a href="#">debug p42</a>	v	run next (skip line and run)
dbq	<a href="#">debug p42</a>	v	queries suspension mode (set by dbr)

dbr	<a href="#">debug p42</a>	v	reset, set suspension mode (0=disable, 1=enable)
dbret	<a href="#">debug p42</a>	v	exit and return argument
dbrr	<a href="#">debug p42</a>	v	re-run with specified arguments
dbrrx	<a href="#">debug p42</a>	v	re-run with specified executed arguments
dbrun	<a href="#">debug p42</a>	v	run again (from current stop)
dbs	<a href="#">debug p42</a>	v	display stack
dbsig	<a href="#">debug p42</a>	v	signal error
dbsq	<a href="#">debug p42</a>	v	stop query
dbss	<a href="#">debug p42</a>	v	stop set
dbst	<a href="#">debug p42</a>	v	returns stack text
dbstack	<a href="#">debug p42</a>	v	displays call stack with header
dbstk	<a href="#">debug p42</a>	v	call stack
dbstop	<a href="#">debug p42</a>	v	set stops on all lines in namelist
dbstopme	<a href="#">debug p42</a>	v	set stops on current definition if y.
dbtrace	<a href="#">debug p42</a>	v	trace control
ddcheck	<a href="#">dd p41</a>	v	check response, display any error message
ddcnm	<a href="#">dd p41</a>	v	column names selected by ddsel
ddcnt	<a href="#">dd p41</a>	v	rowcount of last ddsql command
ddcol	<a href="#">dd p41</a>	v	column names in the table
ddcom	<a href="#">dd p41</a>	v	commit transaction
ddcon	<a href="#">dd p41</a>	v	connect to data source
dddis	<a href="#">dd p41</a>	v	disconnect from data source
ddend	<a href="#">dd p41</a>	v	end sql statement started with ddsel
dderr	<a href="#">dd p41</a>	v	error info on last command (name, source, warning, msg)

ddfch	<a href="#">dd p41</a>	v	as ddfet, but with data in columns
ddfet	<a href="#">dd p41</a>	v	fetch next record from selected data
ddrbk	<a href="#">dd p41</a>	v	rollback transaction
ddsel	<a href="#">dd p41</a>	v	prepare and execute sql statement (selection)
ddsql	<a href="#">dd p41</a>	v	prepare, execute, and end sql statement
ddsrc	<a href="#">dd p41</a>	v	data source names available from ODBC manager
ddtbl	<a href="#">dd p41</a>	v	selection handle for tables in data source
ddtrn	<a href="#">dd p41</a>	v	begin transaction
deal	<a href="#">random p64</a>	v	deal x. items from y. (no repetition)
dealx	<a href="#">random p64</a>	v	deal x. indices from list y. (no repetition)
deb	<a href="#">strings p71</a>	v	delete extra blanks
def	<a href="#">stdlib p70</a>	c	: (explicit definition)
default	<a href="#">misc p55</a>	v	set default value
define	<a href="#">stdlib p70</a>	a	: 0 (explicit definition script form)
delstring	<a href="#">strings p71</a>	v	delete occurrences of x. from y.
destroy	<a href="#">jselect p51</a>	v	destroy select form
detab	<a href="#">convert p37</a>	v	remove tab stops
dev	<a href="#">statfns p69</a>	v	deviation from mean
dfh	<a href="#">convert p37</a>	v	decimal from hex
dfr	<a href="#">trig p74</a>	v	degrees from radians
diff	<a href="#">misc p55</a>	v	second differences
dir	<a href="#">dir p43</a>	v	directory listings
dircompare	<a href="#">dir p43</a>	v	compare files in directories
dirfind	<a href="#">dir p43</a>	v	find name in directory



dirpath	<a href="#">dir p43</a>	v	directory paths
dirs	<a href="#">dir p43</a>	v	browse files in directory
dirss	<a href="#">dir p43</a>	v	directory string search
dirssrplc	<a href="#">dir p43</a>	v	directory string search and replace
dirtree	<a href="#">dir p43</a>	v	get filenames in directory tree
dirused	<a href="#">dir p43</a>	v	get count and space of files in directory tree
discreterand	<a href="#">statdist p68</a>	v	random numbers in a discrete distribution
dlb	<a href="#">strings p71</a>	v	delete leading blanks
dltb	<a href="#">strings p71</a>	v	delete leading and trailing blanks
dltps	<a href="#">strings p71</a>	v	delete multiple leading and trailing blanks
do	<a href="#">stdlib p70</a>	v	name for ".
drawbox	<a href="#">write p79</a>	v	draw box (outline or solid)
drawgraph	<a href="#">write p79</a>	v	draw graph (from wmf file)
drawline	<a href="#">write p79</a>	v	draw line
drop	<a href="#">stdlib p70</a>	v	name for }.
dropafter	<a href="#">strings p71</a>	v	drop after x. in y.
dropto	<a href="#">strings p71</a>	v	drop to x. in y.
dstat	<a href="#">statfns p69</a>	v	descriptive statistics
dtb	<a href="#">strings p71</a>	v	delete trailing blanks
dtbs	<a href="#">strings p71</a>	v	delete multiple trailing blanks in text
dyad	<a href="#">stdlib p70</a>	n	integer 4
each	<a href="#">stdlib p70</a>	a	each (&.>)
empty	<a href="#">stdlib p70</a>	v	return empty result (i.0 0)
endpage	<a href="#">write p79</a>	v	end page
endprint	<a href="#">write p79</a>	v	end print job

erase	<a href="#">stdlib p70</a>	v	erase namelist
every	<a href="#">stdlib p70</a>	a	every (&>)
expand	<a href="#">stdlib p70</a>	v	boolean expand
expandby	<a href="#">format p46</a>	c	expand data with a given value
expandn	<a href="#">format p46</a>	v	expand data at every nth item
exponentialrand	<a href="#">statdist p68</a>	v	random numbers in an exponential distribution
EAV	<a href="#">stdlib p70</a>	n	ascii 255 character
fappend	<a href="#">files p45</a>	v	append text to file
fappends	<a href="#">files p45</a>	v	append string to file
fc	<a href="#">dll p44</a>	v	float conversion
fcompare	<a href="#">compare p36</a>	v	compare two text files
fcopynew	<a href="#">files p45</a>	v	copies files if changed
fcopynews	<a href="#">files p45</a>	v	copies files as strings if changed
fdir	<a href="#">files p45</a>	v	file directory
ferase	<a href="#">files p45</a>	v	erases a file
fetch	<a href="#">stdlib p70</a>	v	name for {::
fexist	<a href="#">files p45</a>	v	test if a file exists
fh	<a href="#">dll p44</a>	v	free header
firstones	<a href="#">parts p60</a>	v	first 1's in partition
fitrect01	<a href="#">isigraph p48</a>	v	fit rectangle data to 0 1
fitrect11	<a href="#">isigraph p48</a>	v	fit rectangle data to _1 1
fit01	<a href="#">isigraph p48</a>	v	fit data to range 0 1
fit11	<a href="#">isigraph p48</a>	v	fit data to range _1 1
flatten	<a href="#">format p46</a>	v	flatten array to a character string

fmt	<a href="#">format p46</a>	v	format a numeric matrix, various format specs
fmtlines	<a href="#">publish p63</a>	v	format lines of J code
fmtscript	<a href="#">publish p63</a>	v	format script
fmtscriptto	<a href="#">publish p63</a>	v	format script
fold	<a href="#">format p46</a>	v	fold text to width
foldpara	<a href="#">text p73</a>	v	fold single paragraph
foldtext	<a href="#">text p73</a>	v	fold text to given width
fread	<a href="#">files p45</a>	v	read file
freadr	<a href="#">files p45</a>	v	read records from flat file
freads	<a href="#">files p45</a>	v	read file as string
freplace	<a href="#">files p45</a>	v	replace text in file
freqcount	<a href="#">statfns p69</a>	v	frequency count
fselect	<a href="#">files p45</a>	v	file selection dialog
fsize	<a href="#">files p45</a>	v	return file size
fss	<a href="#">files p45</a>	v	file string search
fssrplc	<a href="#">files p45</a>	v	file string search and replace
fstamp	<a href="#">files p45</a>	v	returns file timestamp
fstringreplace	<a href="#">strings p71</a>	v	file string replace
fview	<a href="#">files p45</a>	v	view file
fwrite	<a href="#">files p45</a>	v	write text to file
fwrites	<a href="#">files p45</a>	v	write string to file
FF	<a href="#">stdlib p70</a>	n	formfeed character
FIXFONT	<a href="#">sysenv p72</a>	n	fixed space font (used in session windows)
gammarand	<a href="#">statdist p68</a>	v	random numbers in a gamma distribution
gbitmap	<a href="#">isigraph p48</a>	v	bitmap viewer using isigraph control

gbrush	<a href="#">isigraph p48</a>	v	set brush color
gclear	<a href="#">isigraph p48</a>	v	clear graphics window
gdarc	<a href="#">graph p47</a>	v	draw arc
gdarc01	<a href="#">graph p47</a>	v	draw arc in 0 0 1 1
gdchord	<a href="#">graph p47</a>	v	draw chord
gdchord01	<a href="#">graph p47</a>	v	draw chord in 0 0 1 1
gdcolor	<a href="#">graph p47</a>	v	set color
gddraw	<a href="#">graph p47</a>	v	wrapper for draw methods
gdellipse	<a href="#">graph p47</a>	v	draw ellipse
gdellipse01	<a href="#">graph p47</a>	v	draw ellipse in 0 0 1 1
gdlines	<a href="#">graph p47</a>	v	draw lines
gdlines01	<a href="#">graph p47</a>	v	draw lines in 0 0 1 1
gdopen	<a href="#">graph p47</a>	v	open/clear graphics window
gdpen	<a href="#">graph p47</a>	v	set pen size and style
gdpencolor	<a href="#">graph p47</a>	v	set pen color
gdpie	<a href="#">graph p47</a>	v	draw pie-shaped wedge
gdpie01	<a href="#">graph p47</a>	v	draw pie-shaped wedge in 0 0 1 1
gdpixel	<a href="#">graph p47</a>	v	draw pixel
gdpixel01	<a href="#">graph p47</a>	v	draw pixel in 0 0 1 1
gdpolygon	<a href="#">graph p47</a>	v	draw
gdpolygon01	<a href="#">graph p47</a>	v	draw in 0 0 1 1
gdirect	<a href="#">graph p47</a>	v	draw rectangle
gdirect01	<a href="#">graph p47</a>	v	draw rectangle in 0 0 1 1
gdroundr	<a href="#">graph p47</a>	v	rounded rectangle
gdroundr01	<a href="#">graph p47</a>	v	rounded rectangle in 0 0 1 1
gdshow	<a href="#">graph p47</a>	v	show graphics

geomean	<a href="#">statfns p69</a>	v	geometric mean
getdate	<a href="#">dates p40</a>	v	get date from character string
gethue	<a href="#">viewmat p76</a>	v	generate color from color set
gfit	<a href="#">isigraph p48</a>	v	fit data to graphics window
gh	<a href="#">dll p44</a>	v	allocate header
glines	<a href="#">isigraph p48</a>	v	display line connecting points
gopen	<a href="#">isigraph p48</a>	v	open graphics window
gpen	<a href="#">isigraph p48</a>	v	set pen color [;width,style]
gpolygon	<a href="#">isigraph p48</a>	v	draw polygon given points
grayscale	<a href="#">isigraph p48</a>	v	generate grayscale
grgb	<a href="#">isigraph p48</a>	v	set color values
groupndx	<a href="#">numeric p58</a>	v	group indices of y. in x.
gscale	<a href="#">isigraph p48</a>	v	scale from (-1,1) to (0,1000)
gshow	<a href="#">isigraph p48</a>	v	{parent} gshow graph
harmean	<a href="#">statfns p69</a>	v	harmonic mean
hex	<a href="#">convert p37</a>	a	create verb for hex calculation
hexdump	<a href="#">format p46</a>	v	show text as hex and ascii characters
hfd	<a href="#">convert p37</a>	v	hex from decimal
hue	<a href="#">bmp p32</a>	v	generate color from color set
hue	<a href="#">isigraph p48</a>	v	generate color from color set
hueRGB	<a href="#">isigraph p48</a>	v	generate color from RGB color se
ic	<a href="#">dll p44</a>	v	integer conversion
indegrees	<a href="#">trig p74</a>	a	convert function to use degrees:
index	<a href="#">misc p55</a>	v	index where result is _1 if not found, instead of #x.
inrange	<a href="#">validate p75</a>	v	(low, high) inrange data

int01	<a href="#">numeric p58</a>	v	interval in n steps from 0 to 1 (= steps 0 1,n)
inverse	<a href="#">stdlib p70</a>	a	inverse (^:_1)
isbalanced	<a href="#">validate p75</a>	v	pair isbalanced string
isboolean	<a href="#">validate p75</a>	v	data is all 0 or 1
isboxed	<a href="#">validate p75</a>	v	is boxed
ischaracter	<a href="#">validate p75</a>	v	data is character
iscomplex	<a href="#">validate p75</a>	v	data is complex
iscounter	<a href="#">validate p75</a>	v	data is non-negative integer (counting number)
isdate	<a href="#">validate p75</a>	v	is date (as yyyy mm dd)
isinteger	<a href="#">validate p75</a>	v	data is all integer
ismatrix	<a href="#">validate p75</a>	v	data is a matrix
isnumeric	<a href="#">validate p75</a>	v	data is numeric
isotimestamp	<a href="#">dates p40</a>	v	format time stamps as: 2000-05-23 16:06:39.268
isreal	<a href="#">validate p75</a>	v	data is all real
isscalar	<a href="#">validate p75</a>	v	data is a scalar
isunicode	<a href="#">validate p75</a>	v	data is unicode
isunique	<a href="#">validate p75</a>	v	data has no duplicates
isvector	<a href="#">validate p75</a>	v	data is a vector
IFCONSOLE	<a href="#">sysenv p72</a>	n	if a console front end
IFJAVA	<a href="#">sysenv p72</a>	n	if a Java front end
IFUNIX	<a href="#">sysenv p72</a>	n	if UNIX
IFWINCE	<a href="#">sysenv p72</a>	n	if Windows CE
IFWINNT	<a href="#">sysenv p72</a>	n	if Windows NT,2000,XP
IFWIN32	<a href="#">sysenv p72</a>	n	if Windows (9x,ME,NT,2000,XP)

jaddonpath	<a href="#">sysenv p72</a>	v	adds path to addons
jappend	<a href="#">jfiles p49</a>	v	append to jfile, (<i.2 3) jappend 'dat'
jconfigpath	<a href="#">sysenv p72</a>	v	adds path to user configuration files
jcreate	<a href="#">jfiles p49</a>	v	create jfile, jcreate 'dat'
jdup	<a href="#">jfiles p49</a>	v	duplicate jfile, 'new' jdup 'dat'
jerase	<a href="#">jfiles p49</a>	v	erase jfile, jerase 'dat'
jhostpath	<a href="#">sysenv p72</a>	v	converts path name to use host path separator
join	<a href="#">misc p55</a>	v	join boxed items
jread	<a href="#">jfiles p49</a>	v	read jfile, jread 'dat';2
jreplace	<a href="#">jfiles p49</a>	v	replace in jfile, ('new value';123) jreplace 'dat';2 5
jsize	<a href="#">jfiles p49</a>	v	return size of jfile, jsize 'dat'
jsystemdefs	<a href="#">sysenv p72</a>	v	loads appropriate netdefs or hostdefs
jssystempath	<a href="#">sysenv p72</a>	v	adds path to J system directory
jtemppath	<a href="#">sysenv p72</a>	v	adds path to directory for temporary files
juserpath	<a href="#">sysenv p72</a>	v	adds path to user's directory
JIJX	<a href="#">sysenv p72</a>	n	1 if -jijx parameter (i.e. don't create ijx window)
kcreate	<a href="#">kfiles p53</a>	v	create file
kdir	<a href="#">kfiles p53</a>	v	keyword directory
kerase	<a href="#">kfiles p53</a>	v	erase file
keycreate	<a href="#">keyfiles p52</a>	v	create file
keydir	<a href="#">keyfiles p52</a>	v	keyword directory
keydrop	<a href="#">keyfiles p52</a>	v	drop keywords
keyerase	<a href="#">keyfiles p52</a>	v	erase file
keyread	<a href="#">keyfiles p52</a>	v	read data

keyreadx	<a href="#">keyfiles p52</a>	v	read extra data
keywrite	<a href="#">keyfiles p52</a>	v	write data
keywritex	<a href="#">keyfiles p52</a>	v	write extra data
key_new	<a href="#">keyfiles p52</a>	v	make new components
kread	<a href="#">kfiles p53</a>	v	read data for keyword
kurtosis	<a href="#">statfns p69</a>	v	kurtosis
kwrite	<a href="#">kfiles p53</a>	v	write date for keyword
lastones	<a href="#">parts p60</a>	v	last 1's in partition
leaf	<a href="#">stdlib p70</a>	a	leaf (L:0)
lfp	<a href="#">parts p60</a>	v	lengths from partition
linsert	<a href="#">numeric p58</a>	v	linear insert x. (default 2) steps into y.
list	<a href="#">stdlib p70</a>	v	list data formatted in columns
ljust	<a href="#">strings p71</a>	v	left justify
lrep	<a href="#">myutil p56</a>	v	linear representation
lsfit	<a href="#">statfns p69</a>	v	least-squares fit
LF	<a href="#">stdlib p70</a>	n	linefeed character
makeframe	<a href="#">write p79</a>	v	make a frame
map	<a href="#">jmf p50</a>	v	map a file
max	<a href="#">statfns p69</a>	v	maximum
mean	<a href="#">statfns p69</a>	v	arithmetic mean
median	<a href="#">statfns p69</a>	v	median
mema	<a href="#">dll p44</a>	v	memory allocate
memf	<a href="#">dll p44</a>	v	memory free
memr	<a href="#">dll p44</a>	v	memory read
memshare	<a href="#">jmf p50</a>	v	share memory with a process
memshareclose	<a href="#">jmf p50</a>	v	close memory shared with memshare



memw	<a href="#">dll p44</a>	v	memory write
mfv	<a href="#">convert p37</a>	v	matrix from vector
midpt	<a href="#">statfns p69</a>	v	index of midpoint
min	<a href="#">statfns p69</a>	v	minimum
monad	<a href="#">stdlib p70</a>	n	integer 3
nameclass	<a href="#">stdlib p70</a>	v	name for 4!:0
namelist	<a href="#">stdlib p70</a>	v	name for 4!:1
names	<a href="#">stdlib p70</a>	v	formatted namelist
nappend	<a href="#">nfiles p57</a>	v	append to file
nc	<a href="#">stdlib p70</a>	v	name for 4!:0
nfmt	<a href="#">format p46</a>	v	simple numeric formatter
nl	<a href="#">stdlib p70</a>	v	selective namelist
normalprob	<a href="#">statdist p68</a>	v	probability of success in normal distribution
normalrand	<a href="#">statdist p68</a>	v	random numbers in a standard normal distribution,
noun	<a href="#">stdlib p70</a>	n	integer 0
nread	<a href="#">nfiles p57</a>	v	read file
nubcount	<a href="#">misc p55</a>	v	nub + count
nwrite	<a href="#">nfiles p57</a>	v	write to file
on	<a href="#">stdlib p70</a>	c	name for @:
OR	<a href="#">dll p44</a>	v	bitwise OR ( )
pack	<a href="#">pack p59</a>	v	package namelist
packlocale	<a href="#">pack p59</a>	v	package locales
partition	<a href="#">parts p60</a>	v	partition items (1 marks new item)
pathname	<a href="#">misc p55</a>	v	split DOS name into path;name

pcompare	<a href="#">pack p59</a>	v	compare two packages
pd	<a href="#">plot p61</a>	v	plot driver
pdef	<a href="#">pack p59</a>	v	package define
perm	<a href="#">statfns p69</a>	v	permutations of size y.
pex	<a href="#">pack p59</a>	v	remove namelist from package
pfl	<a href="#">parts p60</a>	v	partition from lengths
pget	<a href="#">pack p59</a>	v	return value of name in package
pi	<a href="#">trig p74</a>	n	pi
pick	<a href="#">stdlib p70</a>	v	pick (>@ { )
plot	<a href="#">plot p61</a>	v	standard plot
poissondist	<a href="#">statdist p68</a>	v	discrete values for poisson distribution
poissonprob	<a href="#">statdist p68</a>	v	probability of success in poisson distribution
poissonrand	<a href="#">statdist p68</a>	v	random numbers in a poisson distribution
polygon	<a href="#">isigraph p48</a>	v	vertices of a regular polygon
pps	<a href="#">myutil p56</a>	v	set print precision
prep	<a href="#">myutil p56</a>	v	parenthesized representation
preverse	<a href="#">parts p60</a>	v	partitioned reverse
preview	<a href="#">write p79</a>	v	preview page
print	<a href="#">print p62</a>	v	print text
printfile	<a href="#">print p62</a>	v	print file
printfile2	<a href="#">print p62</a>	v	print file in 2-up mode
print2	<a href="#">print p62</a>	v	print text in 2-up mode
prompt	<a href="#">misc p55</a>	v	prompt for input
psel	<a href="#">pack p59</a>	v	select namelist from package
pset	<a href="#">pack p59</a>	v	merge new into old

psort	<a href="#">parts p60</a>	v	partioned sort
psum	<a href="#">parts p60</a>	v	partioned sum
psumscan	<a href="#">parts p60</a>	v	partioned sumscan
publish	<a href="#">publish p63</a>	v	interactive publisher
PROFILE	<a href="#">sysenv p72</a>	n	name of profile file
PROFONT	<a href="#">sysenv p72</a>	n	proportional font (used in forms)
quote	<a href="#">convert p37</a>	v	quote text
randomize	<a href="#">numeric p58</a>	v	sets a random value into random link
randomize	<a href="#">random p64</a>	v	sets a random value into random link
rand01	<a href="#">random p64</a>	v	generate y. random numbers in interval (0,1)
rand11	<a href="#">random p64</a>	v	generate y. random numbers in interval (_1,1)
range	<a href="#">numeric p58</a>	v	range from a to b [in steps of c]
readbmp	<a href="#">bmp p32</a>	v	read bitmap file, returning RGB data
readbmphdr	<a href="#">bmp p32</a>	v	read header from bitmap file
readcsv	<a href="#">csv p39</a>	v	reads csv file into a boxed array
recur	<a href="#">numeric p58</a>	v	solves recurrence $r(i)=a(i)+r(i-1)*m(i-1)$
regression	<a href="#">statfns p69</a>	v	multiple regression
rfd	<a href="#">trig p74</a>	v	radians from degrees
rjust	<a href="#">strings p71</a>	v	right justify
rotate	<a href="#">isigraph p48</a>	v	rotate angle by given amount
round	<a href="#">numeric p58</a>	v	round y. to nearest x. (e.g. 1000 round 12345)
rounddist	<a href="#">numeric p58</a>	v	round y. to nearest x. preserving total
roundint	<a href="#">numeric p58</a>	v	round to nearest integer
rows	<a href="#">stdlib p70</a>	a	rows ("1)

rplc	<a href="#">strings p71</a>	v	replace characters in text string
ruler	<a href="#">format p46</a>	v	formatted ruler
runindices	<a href="#">parts p60</a>	v	indices from run lengths
runlengths	<a href="#">parts p60</a>	v	lengths of each run
rxall	<a href="#">regex p65</a>	v	regex equivalent of { (all matches)
rxapply	<a href="#">regex p65</a>	v	apply verb to pattern
rxcomp	<a href="#">regex p65</a>	v	compile pattern
rxcut	<a href="#">regex p65</a>	v	cut string into nomatch/match list
rxeq	<a href="#">regex p65</a>	v	regex equivalent of -:
rxerror	<a href="#">regex p65</a>	v	last regex error message
rxE	<a href="#">regex p65</a>	v	regex equivalent of E.
rxfirst	<a href="#">regex p65</a>	v	regex equivalent of {.@{ (first match)
rxfree	<a href="#">regex p65</a>	v	free pattern handles
rxfrom	<a href="#">regex p65</a>	v	matches from string
rxhandles	<a href="#">regex p65</a>	v	list pattern handles
rxin	<a href="#">regex p65</a>	v	regex equivalent of e.
rxindex	<a href="#">regex p65</a>	v	regex equivalent of i.
rxinfo	<a href="#">regex p65</a>	v	info on pattern handles
rxmatch	<a href="#">regex p65</a>	v	single match
rxmatches	<a href="#">regex p65</a>	v	all matches
rxmerge	<a href="#">regex p65</a>	v	replace matches in string
rxrplc	<a href="#">regex p65</a>	v	search and replace
RGB	<a href="#">rgb p66</a>	v	convert between color triples and RGB values
script	<a href="#">stdlib p70</a>	v	load script, cover for 0!:0
scriptd	<a href="#">stdlib p70</a>	v	load script with display, cover for 0!:1

scriptform	<a href="#">misc p55</a>	v	script representation of names
sdaccept	<a href="#">socket p67</a>	v	accept connection
sdasync	<a href="#">socket p67</a>	v	set up async connection for the socket
sdbind	<a href="#">socket p67</a>	v	bind socket
sdcheck	<a href="#">socket p67</a>	v	check socket for errors
sdcleanup	<a href="#">socket p67</a>	v	initialize winsock
sdclose	<a href="#">socket p67</a>	v	close socket
sdconnect	<a href="#">socket p67</a>	v	connect to the socket
sdgethostbyaddr	<a href="#">socket p67</a>	v	returns a name from an address
sdgethostbyname	<a href="#">socket p67</a>	v	returns an address from a name
sdgethostname	<a href="#">socket p67</a>	v	returns host name
sdgetpeername	<a href="#">socket p67</a>	v	returns address this socket is connected to
sdgetsockets	<a href="#">socket p67</a>	v	returns all socket numbers
sdgetsockname	<a href="#">socket p67</a>	v	returns address of this socket
sdinit	<a href="#">socket p67</a>	v	initialize winsock
sdioctl	<a href="#">socket p67</a>	v	read or write socket control information
sdionread	<a href="#">socket p67</a>	v	get number of bytes available for reading socket
sdlisten	<a href="#">socket p67</a>	v	set up listener for the socket
sdrecv	<a href="#">socket p67</a>	v	read data
sdrecvfrom	<a href="#">socket p67</a>	v	read data from
sdselect	<a href="#">socket p67</a>	v	find sockets that need work
sdsend	<a href="#">socket p67</a>	v	send data
sdsendto	<a href="#">socket p67</a>	v	send data
sdsetsockopt	<a href="#">socket p67</a>	v	sets the value of a socket option.
sdsockaddress	<a href="#">socket p67</a>	v	returns address

sdsockerror	<a href="#">socket p67</a>	v	retrieve socket error code
sdssocket	<a href="#">socket p67</a>	v	creates a socket
setrl	<a href="#">random p64</a>	v	set random link
share	<a href="#">jmf p50</a>	v	share a mapped file
show	<a href="#">misc p55</a>	v	show names using linear representation
showmap	<a href="#">jmf p50</a>	v	show all mappings
sign	<a href="#">stdlib p70</a>	a	sign (*)
sin	<a href="#">trig p74</a>	v	sin
sind	<a href="#">trig p74</a>	v	sin in degrees
sinh	<a href="#">trig p74</a>	v	sinh
skewness	<a href="#">statfns p69</a>	v	skewness
smoutput	<a href="#">stdlib p70</a>	v	output to session
snap	<a href="#">myutil p56</a>	v	names snapshot
sort	<a href="#">stdlib p70</a>	v	sort up
spdev	<a href="#">statfns p69</a>	v	sum of products of deviations
split	<a href="#">stdlib p70</a>	v	split head from tail
sqzint	<a href="#">format p46</a>	v	squeeze list of positive integers into short form
sqzrun	<a href="#">format p46</a>	v	squeeze list of numbers into short form
ss	<a href="#">strings p71</a>	v	string search
ssdev	<a href="#">statfns p69</a>	v	sum squares of deviation
startpage	<a href="#">write p79</a>	v	start page
startprint	<a href="#">write p79</a>	v	initialize print job
stddev	<a href="#">statfns p69</a>	v	standard deviation
steps	<a href="#">numeric p58</a>	v	steps from a to b in c steps
stringreplace	<a href="#">strings p71</a>	v	replace characters in text string

subs	<a href="#">misc p55</a>	c	substitution
symget	<a href="#">dll p44</a>	v	get address of locale entry for name
symget	<a href="#">dll p44</a>	v	set array as address
SYSTEM	<a href="#">sysenv p72</a>	n	path to J system directory
table	<a href="#">stdlib p70</a>	a	function table
take	<a href="#">stdlib p70</a>	v	name for {.
takeafter	<a href="#">strings p71</a>	v	take after x. in y.
taketo	<a href="#">strings p71</a>	v	take to x. in y.
tan	<a href="#">trig p74</a>	v	tan
tand	<a href="#">trig p74</a>	v	tan in degrees
tanh	<a href="#">trig p74</a>	v	tanh
time	<a href="#">myutil p56</a>	v	time
timespacex	<a href="#">myutil p56</a>	v	time and space for expressions
timestamp	<a href="#">dates p40</a>	v	format time stamps as: 23 May 1998 16:06:39
timex	<a href="#">myutil p56</a>	v	time expressions
toCRLF	<a href="#">stdlib p70</a>	v	converts character strings to CRLF delimiter
to date	<a href="#">dates p40</a>	v	converts day numbers to dates
todayno	<a href="#">dates p40</a>	v	converts dates to day numbers
toHOST	<a href="#">stdlib p70</a>	v	converts character strings to Host delimiter
toJ	<a href="#">stdlib p70</a>	v	converts character strings to J delimiter (linefeed)
tolist	<a href="#">misc p55</a>	v	convert boxed list to LF delimited list
tolist	<a href="#">myutil p56</a>	v	convert boxed list to LF delimited list
tolower	<a href="#">stdlib p70</a>	v	convert text to lower case
topara	<a href="#">text p73</a>	v	convert text to paragraphs

toss	<a href="#">random p64</a>	v	pick x. items from y. (with repetition)
tossx	<a href="#">random p64</a>	v	pick x. indices from list y. (with repetition)
toupper	<a href="#">stdlib p70</a>	v	convert text to upper case
tree	<a href="#">myutil p56</a>	v	tree representation
tsdiff	<a href="#">dates p40</a>	v	differences between pairs of dates.
tsrep	<a href="#">dates p40</a>	v	timestamp representation as a single number
type	<a href="#">stdlib p70</a>	v	object type
TAB	<a href="#">stdlib p70</a>	n	tab character
TEMP	<a href="#">sysenv p72</a>	n	path to temporary directory for temporary files
unmap	<a href="#">jmf p50</a>	v	unmap a mapped file
unmapall	<a href="#">jmf p50</a>	v	unmap all mapped files
UNAME	<a href="#">sysenv p72</a>	n	name of UNIX o/s
USER	<a href="#">sysenv p72</a>	n	path to user's directory
valdate	<a href="#">dates p40</a>	v	validate dates
var	<a href="#">statfns p69</a>	v	variance
verb	<a href="#">stdlib p70</a>	n	integer 3
vfm	<a href="#">convert p37</a>	v	vector from matrix
viewbmp	<a href="#">bmp p32</a>	v	bitmap viewer using isipicture control
viewmat	<a href="#">viewmat p76</a>	v	view matrix in isigraph control
viewrgb	<a href="#">bmp p32</a>	v	view numeric matrix of RGB data
wcsize	<a href="#">stdlib p70</a>	v	size of execution window
wd	<a href="#">winlib p78</a>	v	main window driver, name for 11!:0
wdbox	<a href="#">winlib p78</a>	v	box wd argument
wdcenter	<a href="#">winlib p78</a>	v	center form on another



wdclipread	<a href="#">winlib p78</a>	v	read clipboard
wdclipwrite	<a href="#">winlib p78</a>	v	write to clipboard
wde	<a href="#">winlib p78</a>	v	as wd but displays error and signals break
wdfit	<a href="#">winlib p78</a>	v	fit form in window
wdget	<a href="#">winlib p78</a>	v	get values from matrix, e.g. wd'q'
wdhandler	<a href="#">winlib p78</a>	v	wd handler
wdinfo	<a href="#">winlib p78</a>	v	information box
wdisparent	<a href="#">winlib p78</a>	v	return 1 if a parent window
wdmenu	<a href="#">menu p54</a>	v	provide a pop-up menu
wdmove	<a href="#">winlib p78</a>	v	position window, relative to side of screen
wdpclose	<a href="#">winlib p78</a>	v	close parent window
wdqshow	<a href="#">winlib p78</a>	v	display result of wdq
wdquery	<a href="#">winlib p78</a>	v	query box
wdreset	<a href="#">winlib p78</a>	v	reset window driver
wdselect	<a href="#">winlib p78</a>	v	selection box
wdstatus	<a href="#">winlib p78</a>	v	put status message on screen
wdview	<a href="#">winlib p78</a>	v	text viewer
weekday	<a href="#">dates p40</a>	v	returns weekday from date, 0=Sunday ... 6=Saturday
winconst	<a href="#">winapi p77</a>	v	look up Windows constants
winset	<a href="#">winapi p77</a>	v	set values of windows constants
win32api	<a href="#">winapi p77</a>	a	look up Win32 API declaration, returning verb
win32apir	<a href="#">winapi p77</a>	a	win32api, except verb returns first element of call result
write	<a href="#">write p79</a>	v	write text (see syntax below)
writebmp	<a href="#">bmp p32</a>	v	write bmp file from RGB data

writcsv	<a href="#">csv p39</a>	v	writes a boxed array to a csv file
xfmt	<a href="#">format p46</a>	v	format extended integers
xtab	<a href="#">print p62</a>	v	remove tab stops from character string
xtabline	<a href="#">print p62</a>	v	remove tab stops
XOR	<a href="#">dll p44</a>	v	bitwise XOR (^)

## Script Listings

Short Name	Full Name
<a href="#">bmp p32</a>	system\packages\graphics\bmp.ijs
<a href="#">colib p33</a>	system\main\colib.ijs
<a href="#">color16 p34</a>	system\packages\color\color16.ijs
<a href="#">colortab p35</a>	system\packages\color\colortab.ijs
<a href="#">compare p36</a>	system\main\compare.ijs
<a href="#">convert p37</a>	system\main\convert.ijs
<a href="#">coutil p38</a>	system\main\coutil.ijs
<a href="#">csv p39</a>	system\packages\files\csv.ijs
<a href="#">dates p40</a>	system\main\dates.ijs
<a href="#">dd p41</a>	system\main\dd.ijs
<a href="#">debug p42</a>	system\main\debug.ijs
<a href="#">dir p43</a>	system\main\dir.ijs
<a href="#">dll p44</a>	system\main\dll.ijs
<a href="#">files p45</a>	system\main\files.ijs
<a href="#">format p46</a>	system\main\format.ijs
<a href="#">graph p47</a>	system\classes\graph\graph.ijs
<a href="#">isigraph p48</a>	system\packages\graphics\isigraph.ijs
<a href="#">jfiles p49</a>	system\packages\files\jfiles.ijs
<a href="#">jmf p50</a>	system\main\jmf.ijs
<a href="#">jselect p51</a>	system\classes\input\jselect.ijs
<a href="#">keyfiles p52</a>	system\packages\files\keyfiles.ijs
<a href="#">kfiles p53</a>	system\packages\files\kfiles.ijs

<a href="#">menu p54</a>	system\packages\winapi\menu.ijs
<a href="#">misc p55</a>	system\main\misc.ijs
<a href="#">myutil p56</a>	system\main\myutil.ijs
<a href="#">nfiles p57</a>	system\packages\files\nfiles.ijs
<a href="#">numeric p58</a>	system\main\numeric.ijs
<a href="#">pack p59</a>	system\main\pack.ijs
<a href="#">parts p60</a>	system\main\parts.ijs
<a href="#">plot p61</a>	system\classes\plot\plot.ijs
<a href="#">print p62</a>	system\packages\print\print.ijs
<a href="#">publish p63</a>	system\packages\publish\publish.ijs
<a href="#">random p64</a>	system\packages\stats\random.ijs
<a href="#">regex p65</a>	system\main\regex.ijs
<a href="#">rgb p66</a>	system\packages\color\rgb.ijs
<a href="#">socket p67</a>	system\main\socket.ijs
<a href="#">statdist p68</a>	system\packages\stats\statdist.ijs
<a href="#">statfns p69</a>	system\packages\stats\statfns.ijs
<a href="#">stdlib p70</a>	system\main\stdlib.ijs
<a href="#">strings p71</a>	system\main\strings.ijs
<a href="#">sysenv p72</a>	system\main\sysenv.ijs
<a href="#">text p73</a>	system\main\text.ijs
<a href="#">trig p74</a>	system\main\trig.ijs
<a href="#">validate p75</a>	system\main\validate.ijs
<a href="#">viewmat p76</a>	system\classes\view\viewmat.ijs
<a href="#">winapi p77</a>	system\packages\winapi\winapi.ijs
<a href="#">winlib p78</a>	system\main\winlib.ijs
<a href="#">write p79</a>	system\packages\print\write.ijs

## **Development Environment**

[Menu Commands p83](#)

[Component Files p84](#)

[Keyed Files p85](#)

[Find in Files p86](#)

[Printing p87](#)

## Menu Commands

Several menu commands invoke J sentences, which typically execute verbs in the j locale. These are as follows:

Menu

Shortcut

Verb

File|New Class

```
filenewform ''
```

File|Print

Ctrl+P

```
fileprint ''
```

Edit|Input Log

Ctrl+D

editinputlog ''

Edit|Find

Ctrl+F

editfind ''

Edit|Find in Files

Ctrl+Shift+F

fif ''

Edit|Configure

config ''

Run|Project Manager

Ctrl+B

projectmanager ''

Run|Debug

Ctrl+K

debug ''

Run|HTML Publish

publish ''

Studio|Labs

lab ''

Studio|Advance



lab 0

Studio|Chapters

lab 1

Studio|Author

lab 2

Studio|Options

lab 3

Studio|Demos

demos ''

{none}

Ctrl+F1

help ''

## Component Files (jfiles)

Jfiles are component `files` for J data. A jfile can be thought of as a boxed list which is stored on file. An element of the boxed list is referred to as a component, and can store a noun of any type, shape or size. File components are numbered sequentially from 0 upwards.

To access the system (assuming you are using the standard profile):

```
load 'jfiles'
```

This populates a locale `jfiles` with utility functions, and defines the following verbs in the `z` locale:

```
jcreate  
jerase  
jappend  
jread  
jreplace  
jdup  
jsize
```

You create a file using `jcreate`.

`jappend` adds new items to the file. Each item added to the file is stored in a new component. Several items can be added to the file at a time.

`jread` reads items from file, and `jreplace` replaces items on file.

`jdup` duplicates a file, and `jsize` returns its size.

You can refer to a file either by its file name, or by its file handle. The default file extension is `.ijf` - this is used if no file extension is given.

You cannot delete components once created. If you need to reclaim space no longer required, either replace the components with empties, or else duplicate the file, copying only the components required.

### *Create file*

#### **jcreate**

create a file. The right argument is the filename. Any existing file is overwritten. The result is 1 if successful, else 0. For example:

```
jcreate 'mydata'
1
```

Note that since the file extension was not specified, this actually creates a file named mydata.ijf.

### *Read and write file*

#### **jappend**

append to file. The left argument is a boxed list, with each item in the list stored in a new file component. An open noun is treated as a single boxed item. The right argument is the filename. The result is the new component numbers created. For example:

```
'header' jappend 'mydata'
0

('rec1';'rec2') jappend 'mydata'
1 2

(< <\1 2 3) jappend 'mydata'
3

(2 2$10 20 30 40) jappend 'mydata'
4
```

#### **jread**

read file. The right argument is the filename, linked with one or more component numbers. The result has the same shape as the component numbers. For example:

```
> jread 'mydata';4
10 20
30 40
```

```
jread 'mydata';i.5
+-----+-----+-----+-----+-----+
|header|rec1|rec2|+-+---+-----+|10 20| | | | |
|      |   |   ||1|1 2|1 2 3||30 40|
|      |   |   |+-+---+-----+|      |
+-----+-----+-----+-----+-----+
```

```
jread 'mydata';i.2 2
+-----+-----+
|header|rec1      |
+-----+-----+
|rec2  |+-+---+-----+| | | | |
|      ||1|1 2|1 2 3||
|      |+-+---+-----+|
+-----+-----+-----+
```

## jreplace

replace in file. The result is the components replaced. For example:

```
(1000;'abcde') jreplace 'mydata';1 2
1 2
```

```
jread 'mydata';i.3
+-----+-----+-----+
|header|1000|abcde|
+-----+-----+-----+
```

The left argument is reshaped if necessary to match the components in the right argument. For example, the following replaces components 1-3, each with the word 'reserved':

```
'reserved' jreplace 'mydata';1 2 3
```

1 2 3

### *Utilities*

#### **jdup**

duplicate file. The left argument is the new filename; if elided, the file is duplicated in place. The right argument is the source filename, optionally linked with one or more component numbers to be copied to the new file, in the order given. By default, the entire file is duplicated. The result is the number of components written. For example:

```
'newdata' jdup 'mydata'
```

5

```
'newdata' jdup 'mydata';2 0 1
```

3

#### **jsize**

size of file, as 4 numbers:

starting component number (0)

number of components

length of file in bytes (same as result of 1! : 4)

amount of free space that could be recovered by duplicating the file

For example:

```
jsize 'newdata'
```

0 5 1312 0

#### **jerase**

erase file, for example:

```
jerase 'newdata'
```

1

### *File handles*

You can refer to a file either by its filename, or by its file handle if you have already opened the file. For most purposes we recommend using the filename. However, if you have a great deal of file activity, you may find it faster to open the file first, then use the file handle; this means the system does not have to open and close the file each time it is accessed. The utilities `jopen` and `jclose` in the `jfiles` locale can be used for this purpose. For example:

```
h=. jopen_files_ 'mydata'

... process file using handle h...
    jclose_files_ h
1
```

### *File structure*

Each file is structured as a header record, followed by data. The header record is as follows:

- [0] version
- [1] starting component
- [2] number of components
- [3] file length
- [4] directory pointer
- [5] freelist pointer
- [6] sequence number

### *Freelist*

Each component is stored in its 3!:1 representation in a space that is a power of 2. This means that on average, the representation fills 75% of the space allocated, and allows some space for growing replaces. If the space required on replacement is less than half the space allocated, then the balance is freed up.

If you replace a component with a noun of smaller size, then this may result in

some unused space. The system keeps track of this in the freelist, and attempts to reuse it where possible. The total free space available is given in the fourth element of the result of `jsize`, and this is the space that would be freed by duplicating the file using `jdup`.



## Keyed Files (keyfiles)

A keyed file is a J component file in which the components are accessed using keywords.

A keyword may be any character string.

Load the keyed file system with:

```
load 'keyfiles'
```

This defines the main functions:

keycreate

create file

keydir

keyword directory

keyerase

erase file

keyread

read data for keyword

keywrite

write data for keyword

read: if keyword not found, return " else return value

write: if data=" then keyword is deleted

### Examples:

```
keycreate 'mydata'
1
    'Peter Rabbit' keywrite 'mydata';'name'
    'Lake District' keywrite 'mydata';'loc'

keydir 'mydata'
loc  name

keyread 'mydata';'loc'
Lake District
```

## Find in Files

Ctrl+Shift+F or menu item Edit|Find in Files starts the Find in Files utility

You can search for simple text, or more complicated patterns with a Regular expression. The Regex button selects the regular expression search; this is automatically enabled when you use the Insert button to insert components that match certain characters, strings, or J-related strings. You can also search in specific contexts, such as assignment or use of names.

If you have projects created by the Project Manager, Find in Files loads with the current Project files in the search path. You can select other projects to search.

If Find in Files is open and you change the current Project in Project Manager, the change is not reflected immediately in Find in Files. To change to the new Project, select menu Options|Refresh Project.

Otherwise, you can search through folders. In this case, you can specify the file types to search, and the folders to search. Customize these by pressing the buttons next to the selection boxes.

Run a search by pressing Enter or Find.

Open a file by putting the caret on a line and pressing Open.

## Printing

Printing from J is handled by the print package. Load with:

```
load 'print'
```

This defines utilities in the jprint locale, plus the following user definitions:

print

print text

printfile

print file

print2

print text in 2-up mode

printfile2

print file in 2-up mode

**verb: print**

form: opt print data

where opt is an optional list separated by semicolons:

ascii [1|0]

set ascii box-drawing characters on|off

font fontspec

set font

fontsize points

set fontsize

filename

set filename text

fit

fit text to page width

header text

set header text

footer text

set footer text

land | landscape

set landscape mode (default is portrait)

ruler

add ruler to top of each page (use with fixed pitch font)

**verb: printfile**

form: opt printfile files

Options are as for print, except the filename is set automatically

**verbs : print2 printfile2**

These verbs are the same as print and printfile, except that the page is printing in landscape mode, 2 pages per sheet. For example, this is good for printing script

files.

Options are as for print, except that fit, land and landscape are ignored, and there is a new option:

cols

set columns, default 80

Default fonts and options can be set in menu Edit|Configure|Print.

The font for 2-up printing should be fixed pitch and around 7-7.5 point. "Lucida Console" 7.25 bold is good if available - use oem if your printer supports it, else ansi.

The ruler option uses the current session box-drawing characters. Choose a box-drawing font, or include the 'ascii' option.

Examples:

```
'font arial 12;land;footer just testing' print i.3 4 5

'ascii;font "courier new" 14' print ;/i.3 4 5

'fontsize 7.5;land;ruler' printfile
'system\examples\data\orders.prn'

printfile2 'system\main\pack.ijs'
```

## **Debug**

[Overview p89](#)

[Verbs p90](#)

[Commands p91](#)

[Stops p92](#)

[Watch p93](#)

## Overview

Debug is an extension to the facilities provided by the debug foreign conjunction. After Debug is turned on, then if execution is halted because of a stop or execution error, the Debug window shows the current execution environment.

Panels in the Debug window show the code being executed, the stack, and values of various names:

- any watch names
- the arguments of the current definition
- any names on the current line of code.

For an introduction to Debug, see the Debug lab.

For information on the debug foreign conjunction, see the Dictionary entry for the `13!:x` family, and script `system\main\debug.ijs`.

## Load Debug

To load Debug, use menu Run|Debug..., or press Ctrl-K when the J session has focus.

This loads the `debug.ijs` script, displays the Debug window, enables suspension (`dbr 1`), and sets the debug latent expression (`dblxs`) to run the verb `debug` in the



jdebug locale.

Then run your program. If execution is halted because of a stop or execution error, the execution state is shown in the Debug window. Several commands let you step through the execution session.

To turn Debug off and disable suspension, close the Debug window.

## Debug Settings

Debug is active when suspension is on, and the debug latent expression is set. The Debug window is displayed when Debug is activated, so this provides a visual indication that Debug is active.

However, if you change either of these directly, for example, by entering:

```
dbr 0
```

then Debug will no longer be active, even if the Debug window is still displayed. You can reactivate Debug by pressing Ctrl-K.

## Verbs

Debug is an addition to the definitions in script system\main\debug.ijs, which can be loaded as follows:

```
load 'debug'
```

This script is loaded when you press Ctrl-K to run Debug.

The script defines:

dbg	turns Debug on (dbg 1) or off (dbg 0)
dblocals	displays local names on the stack
dbstack	displays the stack
dbstop	sets stops on all lines in a list of names
dbstopme	sets stops in current definition

## Verb dbg

To turn Debug on, enter:

```
dbg 1          NB. same as Ctrl-K
```

To turn Debug off and disable suspension, enter:

```
dbg 0          NB. same as closing Debug window
```

## Verb dblocals

Displays local names on the stack, for example:

```
dblocals''
+---+-----+
| jmp | +---+---+ | | | |
|     | |a |123| |
|     | +---+---+ |
|     | |y.|   | |
|     | +---+---+ |
+---+-----+
| run | +---+---+ | | | |
|     | |c |3 |   |
|     | +---+---+ |
|     | |y.|   | |
|     | +---+---+ |
+---+-----+
```

## Verb dbstack

Displays the current stack, for example:

```
db 1
plus=. +
plus / 'abc'
dbstack''
+---+---+---+---+---+---+---+
|name|en|ln|nc|args |locals|susp|
+---+---+---+---+---+---+---+
|plus|3 |0 |3 | +---+ |      |* | | | | |
|    | | | | | |b|c| |      | |
|    | | | | | +---+ |      | |
+---+---+---+---+---+---+---+
```

+-----+---+---+---+-----+-----+-----+

## Verb dbstop

This is a convenient way of setting stops on all lines in a list of names, for example:

```
dbstop 'accpay intm vrep'
```

is equivalent to:

```
dbss (dbsq''), 'accpay *:*;intm *:*;vrep *:*'
```

Using dbstop does not change any other stop settings, except that to clear all stops, enter:

```
dbstop''
```

## Verb dbstopme

This is a convenient way of setting stops on the current definition. If the argument is non-zero (with same meaning as the if. control word) and suspension is on, then all lines in the current definition are stopped, for example:

```
foo=: 3 : 0
dbstopme y.>2      NB. stop foo if y.>2
...
)
```

Using dbstopme does not change any other stop settings.

## Commands

In the code panel of the Debug window:

- > marks the suspended line, same as the initial run line.
- marks the run line, if moved back from the suspended line.
- + marks the run line, if moved forward from the suspended line.
- \* marks a stop on a specific line.
- | marks a stop on all lines.

## Toolbar Commands

Command	Shortcut	Description
---------	----------	-------------

run	F5	continue execution at the run line.
step into	F6	continue execution at the run line, stopping on the next line, or on the first line of any definitions on the run line.
step over	Enter, F7	continue execution at the run line, stopping on the next line, or earlier stop..
step out	F8	continue execution at the run line, stopping on the next line of the calling function. Does not stop again in the current definition, except where stops have been set.
cutback		cuts back stack to calling function.
back		move run line back one line. Does not undo sentences already executed.
refresh		refresh Debug window, and restore run line to suspended line.

forward	move run line forward one line. Does not execute the run line.
stop name	set stops on all lines on name at cursor position, or, if the entire line is selected, on all names on the line. If successful, highlights the name, or entire line of code.
stop manager	manage the stop settings.
watch manager	manage the watch names.
edit	open script window with the current definition.
clear	reset stack and clear the Debug window. Leaves suspension on.
help	This help.

## Other Commands (not on Toolbar)

stop line

F9

toggle stop on line at the cursor position.

»



## Stops

You can set and query stops using the foreign conjunctions `dbss (13!:3)` and `dbsq (13!:2)`, and using the utilities `dbstop` and `dbstopme`.

You can also set and view stops with Debug:

- The *stop name* button in the Debug toolbar sets stops on all lines in the selected name.
- The Debug Stop Manager lets you set stops on any active definition.

For names whose definition has more than one line, Debug distinguishes between stops set on specific lines (marked with `*`) and stops set on all lines (marked with `|`). For one-line definitions, only the "all lines" marker is used.

The namelist in the Stop Manager defaults to the selected definition (if any), and the names on the stack. Select the In Locale checkbox to view names by locale. Place the cursor on a name, and run the Stop Manager toolbar button to view that name.

## Notes

Comment lines are ignored when setting stops.

## Watch

The watch is a list of names that will be displayed at the top of the Debug window values panel. The list is stored in noun `WATCH_jdebug_` .

You can set the watch list from the Debug Watch Manager.

You can also set the watch directly as a character string, or a boxed list of names, for example:

```
WATCH_jdebug_=: 'calc data DEFS'
```

## **Project Manager**

[Overview p95](#)

[Project File p96](#)

[Project Manager p97](#)

[Project Manager Tabs p98](#)

[Building Applications p99](#)

[Project Conventions p100](#)

## Projects Overview

A [project file p96](#) lets you work with applications that are built from several scripts. You can specify which scripts are to be included in the application, which are required for development purposes only, and how the application is to be built.

While the scripts are maintained individually during development, they can be compiled into a single output script for distribution/runtime/installation purposes. Alternatively, you can specify a build script that is run to create the final application, which may consist of several scripts.

The [Project Manager p97](#) lets you work with project files. Run the Project Manager from menu Run|Project Manager... or by pressing Ctrl-B.

Project files distinguish between scripts you write specifically for the project, and library scripts distributed with the system that are needed in the project. The project system automatically includes any files required by the scripts you specify.

Typically all the files in a project, apart from library files, will be stored in a single directory, for example, a subdirectory of user\projects. Within the directory will be a project file, plus the project source scripts.

The project facilities require that you adhere to various naming and coding [conventions p100](#).

## Project File

A project file is a script file with extension .ijp that defines the project. It is created and maintained by the [Project Manager p97](#).

A project file defines the following nouns:

### PRIMARYFILES

Project-specific files, required for the application

### PRIMARYLIBS

Standard libraries needed for the application

### DEVFILES

Project-specific files, required for development only

### DEVLIBS

Standard libraries required for development only

### OTHERFILES

Other files accessible in Project Manager

### TARGETFILE

Target file for the application

### TARGETLOCALE

Target locale

### TARGETHEADER

Header comments written to target file

## TARGETEXTRA

Extra code appended to target file

## BUILD\_OPTS

Build options (numeric list)

## BUILDFILE

Special build file

## NOTES

Free-form notes

## WINDOWS

Project open window positions

## WINDOWSTATE

Project open window state

The distinction between PRIMARY or DEVELOPMENT is that when you create an application from the project, only the primary files are used. The development files are assumed to be required for development purposes only.

FILES refer to files specific to the project, and LIBS to library files that are supplied with the J system. Library files are referenced by their short names. Note that library files `stdlib.ijs`, `colib.ijs` and `winlib.ijs` are assumed to be always available, and therefore do not appear in the selections for library files.

OTHERFILES is an additional list of files that you can access when using Project Manager. For example, these include the project file itself, the target file (if specified), and any other files that you want to access easily, but not load with the project.

The project system uses the `jproject` locale for storing the project definitions.

However, the project scripts are loaded into the TARGETLOCALE or base locale if not given.

TARGETFILE is the file created when building the final application. If TARGETLOCALE is given, the locale is set after any library files are loaded. TARGETEXTRA is an optional line of code to be appended to the targetfile, typically to run the application when it is loaded. TARGETHEADER comments are lines prefixed by NB. that are added to the top of the file. Use this for the file description.

BUILD\_OPTS configure the targetfile. BUILDFILE is an optional script file run when building the application.

A project file is a plain text file and can be edited directly, as long as you preserve the same names. You should not add new definitions to the file, since the Project Manager will overwrite them when you next use it. Instead, add any new definitions to other script files in the project.

## Project Manager

The Project Manager lets you work with project files. Run the Project Manager from menu Edit|Project Manager... or by pressing Ctrl-B.

From the Project Manager dialog, open a project file by either:

- selecting from the Project list
- running menu File|New...
- running menu File|Recent...

The Project list shows all files with extension .ijp found in the Look In folder and its subfolders. If the project filename is the same as its folder name, for example, mygraph\mygraph.ijs, then only the filename is shown.

You should set Look In to point to the folders (directories) where you are creating J projects. For example, the default Look In folder is user\projects. You might then create projects in directories such as user\projects\work, user\projects\test etc.

You can define and modify the Look In folders by pressing the >> button.

The menu File|Recent... shows the last 10 projects that have been loaded.

When a project is open, use Load to load the project on top of existing definitions. Select menu Project|Clean Load to reset the system, clearing out existing definitions, and then load the project.

Select Project|Clean Load Primaries to reset the system, and then load the project primary files only; this is useful for testing the final application. The verb loadp can also be used for this purpose.

Use menu Project|Build Options... to specify how the application is to be built. Use menu Project|Build Application to build the application.

## Notes

The Notes editbox in menu File|Notes is for free-form comments. These comments



are stored in the project file only, and do not appear in the final application.

## Project Manager Tabs

### Source

The Source tab lists the files in the project. These are the scripts in the project other than library scripts.

Files in the same directory as the project file are shown without any directory path. Other files have their directory paths.

Use the menu Options|Mark as Dev Only to mark a script as being for development only; its name is then shown followed by (d).

### *Required Files*

If a source script depends on other scripts, for example, if it contains a line of the form

```
load 'source\util\myutil'
```

then the required file will also be shown, with a name followed by (r) or (dr).

This feature is primarily intended for library files. Typically, project source files need not require other files, since it would be simpler to add them explicitly to the source file list.

### Library

The Library tab shows the library files in the project, as well as the library files that are available. Files are shown with their short name only.

As with Source files, use the menu Options|Mark as Dev Only to mark files that are for development only.

Required files have names followed by (r) or (dr). Unlike Source files, some library files are expected to require other files.

### Project

Here you can include any of the following four scripts; use the Add button to add or modify one:

Target	the target file used when Project Manager builds the application
Pre-Build	a script file that is run before Project Manager builds the target file
Post-Build	a script file that is run after Project Manager builds the target file
Test	a script file that is run when you press the Test button

Include a Target file whenever you want Project Manager to build the application for you.

The Pre-Build and Post-Build scripts allow you to customize the behavior of Project Manager's build routine. Also, if you do not want Project Manager to build your application, you can nevertheless define one or both "build" scripts with your own custom build routines.

Include a Test file so that you can test your application by pressing the "Test" button.

### **Other**

The Other tab shows any other files you may want to include in the project, for example, this is a good place to put additional test scripts, that may be loaded by the project's Test script.

## Building Applications

You build the application by clicking the Build button in the Project Manager and Build Options dialogs.

### Include Files

You can choose whether to include or require files when building the targetfile; "include" means the scripts form part of the target file, "require" means there is a line of the form: `script_z_ <'filename'`

written into the target file.

For applications that will be run when the J development system is loaded, you would typically want to require any project libraries, and either include or require project source.

For standalone files, you should require libraries only if you know that the application will be run on your own machine, and so the library files will be available. Otherwise, you should select to include all library files; in this case, any library script referenced by the project will be included in the targetfile.

The standard libraries that are included in standalone applications are:

- colib.ijs
- loadlib.ijs
- stdlib.ijs
- winlib.ijs

The Libraries for J DLL option excludes winlib.ijs from the standard libraries.

### Build Options

The delete comments option removes all comments from your source files, except only any comments that appear on consecutive lines in your first script.

If Load in locale is given as for example, "myloc", the script will include a line:

```
coclass 'myloc'
```

after any standard libraries are loaded, and before the project files are loaded.

If Append to file is given, it is added as the last statement in the application. Use this to run the application when it is loaded.

## Project Conventions

The project facilities require that you adhere to the following conventions:

- File extensions must be:

ordinary scripts

`ijs`

locked scripts

`ijl`

runtime scripts

`ijr`

project files

`ijp`

- Dependencies must be given in script files with lines of the form:

```
load 'dates regex strings'
```

where the verb is either `load`, `require` or `corequire`, and the files are given in a text string. Specifically, the first word on the line must be one of the allowed verbs, and this must be followed by a character string. Blank separators are ignored.

Note that where dependencies are given in the right form, the project system will automatically include the required files in the project.

There are no requirements as to project directories. However, it makes for easier searching if you group the project subdirectories in a small number of directories. For example, the recommended project directory is `user\projects`, and you might then create subdirectories such as `user\projects\mywork`, `user\projects\utils` etc.

Also, you may wish to name project files with the same name as the directory, since this simplifies the display in the Look In list box.

Since directories are searched for project files when you use the Project Manager, you should avoid using directories where there are a large number of subdirectories not being used for projects. For example, you would typically not include projects under the J system subdirectory.

## Locked Scripts (ijl files)

The J development system can load definitions from a locked script file. The names defined by running a locked file can be used normally, but their definitions are hidden from the user. A locked file normally has a file suffix of .ijl. This allows you to use proprietary or protected definitions within the framework of an open J development system. Source is locked with verb 3!:6, for example:

```
dat=. fread 'user\projects\util\myapp.ijs' NB. read source
    dat=. 3!:6 dat                          NB. lock source
    dat fwrite 'user\projects\util\myapp.ijl' NB. create ijl file
```

The text for abc.ijl is encoded and running it defines its verbs, adverbs, and conjunctions as locked. Locked definitions will not be disclosed by the J interpreter. For example, create foo.ijs and foo.ijl with the following definitions:

```
plus =: 4 : 'x.+y.'
mean =: 3 : '(plus/y.)%#y.'
plust=: +
meant=: plust/ % #
```

Load and test as: 0!:1 <'foo.ijs'		0!:1 <'foo.ijl'
3 plus 'abc'		3 plus 'abc'
domain error: plus		domain error: plus
x. +y.		3 plus 'abc'
mean 'abc'		mean 'abc'
domain error: plus		domain error: plus
x. +y.		mean 'abc'
13!:0 [1		13!:0 [1
mean 'abc'		mean 'abc'
domain error: plus		domain error: plus
x. +y.		mean 'abc'
plus[:0]		
13!:1 ''		13!:1 ''
domain error		
x. +y.		
plus[:0]		
( plus/y.)%#y.		
mean[0]		
mean 'abc'		



```
13!:0 [1
plus
4 : 'x.+y.'
meant f.
+ / % #
```

```
| 13!:0 [1
| plus
| plus
| meant f.
| meant
```

## **Window Driver**

[Overview p103](#)

[wd p104](#)

[Window Forms p105](#)

[Event Handlers p106](#)

[wdhandler p107](#)

[Entering Information p108](#)

[Form Locales p109](#)

[Other Message Handlers p110](#)

[Wait p111](#)

[System Events p112](#)

## Window Driver Overview

The J/Windows interface is referred to as the *Window Driver* and provides access under program control to many features of the Windows graphical and operating system environment. It provides essentially two mechanisms for communication:

- you can send instructions and queries to Windows.
- you can react to events signaled by Windows, such as the press of a button on a form.

The verbs `wd` and `wdhandler` defined in script file `winlib.ijs` provide these mechanisms:

- `wd` is used to send instructions and queries to the Window Driver - and is the only way of doing so.
- `wdhandler` is used to react to events signaled by Windows. It is the standard way of doing so, but may be changed if required.

These verbs are defined in the standard profile when you load J.

**wd**

This is defined as:

```
wd = . 11! : 0
```

`wd` is used monadically and takes an argument of a character string of one or more statements, separated by semicolons. These statements are converted by the Window Driver into the appropriate Windows function calls. The result is the result of executing the last statement. Each statement is a command followed by any required parameters.

Commands supported include:

- creation and use of Windows controls to allow user input
- instructions to the J session manager
- DDE and OLE links to other Windows software
- access to custom controls
- information queries

For details and a list of commands see the `wd` Commands help file.

The result of `wd` is the information requested, typically as a character string, or empty if none. For example, the following queries the system metrics used by Windows (screen width, screen height, x logical units and y logical units:

```
wd 'qm'
640 480 8 16
```

This shows a 640 by 480 screen, with the base dialog font (used when creating forms) being 8 pixels wide and 16 pixels high.

The next example creates a dialog box that allows the user to select a font. The result is returned after the user closes the dialog box, and contains a string of items that describe the font selected:

```
wd 'mbfont'
```

```
"Lucida Console" 10 bold
```

In some cases, the result is a character string delimited by LF, which can be formatted by `list`. The following displays all control styles:

```
list wd 'qs'
bs_defpushbutton    bs_leftttext        bs_ownerdraw
cbs_autohscroll     cbs_sort           es_autohscroll
es_autovscroll      es_center          es_lowercase
...
```

An invalid command is signalled as a domain error:

```
wd 'qz'
|domain error
|      wd'qz'
```

After an error, you can run command `qer` to query the error - the result is an error message, following by the index in the command where the error occurred:

```
wd 'qer'
invalid command : 0
```

The following example executes Notepad, assuming it is available on your system - if not, try executing another program.

```
wd 'winexec "\windows\notepad.exe" '
0
```

The next example executes Notepad with the given filename argument (Notepad loads the file).

```
wd 'winexec "\windows\notepad.exe system\main\dates.ijs" '
0
```

In the case of an instruction, the result is empty. For example, the following sets the session manager to tile open windows (assuming there are two or more

windows open), and returns an empty result:

```
$ wd 'smtile'  
0 0
```

## Windows Forms

`wd` is used to create the forms (windows and controls) in a user interface. The easiest way to do this is to use the Form Editor, which builds the appropriate `wd` calls for you.

For a tutorial introduction to the Form Editor, see lab *Form Editor*.

The following example creates a window `mywin`, adds a push button named `pressme`, then shows it as a topmost window (i.e. it stays on top of any other window):

```
wd 'pc mywin;cc pressme button;pshow;ptop'
```

Once it is displayed, you can move the window away from the J session, so that you can see both the session and the window. You can set focus on the J session or in the window, by clicking on them.

If you now click on the button, it depresses, but otherwise nothing seems to happen. In fact, the click on the button causes an *event*. If a corresponding *event handler* is defined, then it is invoked, otherwise the event is ignored.

The following example is an event handler for the button. To try this out, enter the following definition:

```
mywin_pressme_button=: wdinfo bind 'button pressed'
```

Now when you click on the button, this event handler is run, and an information box pops up.

The event handler `mywin_pressme_button` is an ordinary J verb. It was defined using the standard utility `wdinfo` that displays an information message box, together with a specific argument `'button pressed'`. For example, you can run this program from the J session, by entering:

```
mywin_pressme_button''
```

## Event Handlers

An event handler is a verb that is invoked when a Windows event occurs, for example, when a button is pressed. The standard event handling mechanism in J supports three levels of event handler for a given form. If the form name is *abc*, these are:

- `abc_handler` (the *parent* handler)
- `abc_id_class` (e.g. `abc_pressme_button`)
- `abc_formevent` (e.g. `abc_close`)
- `abc_default` (the *default* handler)

When an event occurs, the system searches for an event handler in the order given, and executes the first one it finds. Therefore, for a form *abc*:

- if a verb `abc_handler` is defined, it is executed for every event associated with that form
- if no such verb is defined, and the event is for a specific control, such as button `pressme`, then if a verb for that control such as `abc_pressme_button` is defined, it is executed;
- or if the event is a form event and a verb for that event is defined, such as `abc_close`, then it is executed
- if no such verb is defined, then if `abc_default` is defined, it is executed
- if no event handler verb is found, the event is ignored

Typically, most forms will be written using only the second level of event handler. The other two levels allow the programmer to deal easily with special cases.

We can try this out on the form defined above. First define a default handler:

```
mywin_default=: wdinfo bind 'this is the default'
```

Click on the form to give it focus, then try pressing a function key. Each time, this new default event handler will be executed. However, when you press on the



button, its own event handler is executed.

Now try defining a parent handler:

```
mywin_handler=: wdinfo bind 'this handles all events'
```

Once this is defined, it handles all events from the form.

Note that these event handlers are ordinary verbs and can be defined and modified as required. The search for an appropriate event handler takes place at the time the event occurs. For example, delete the parent handler:

```
erase 'mywin_handler'
```

1

Now the other event handlers will again be used to respond to the form's events.

## wdhandler

This is the verb that provides the mechanism described above. When a Windows event occurs, the system typically invokes the following sentence (but does not show it in the session):

```
wdhandler ' '
```

To demonstrate this, try defining a new `wdhandler` as follows:

```
wdhandler=: wdinfo bind 'my new handler'
```

Now any action you take on the form will invoke this new definition. Typically, you would not want to redefine `wdhandler`, but the fact that you can do so gives you complete control over the way events are handled. To erase your definition and recover the old definition (which is in locale `z`), enter:

```
erase 'wdhandler'
```

How does the standard `wdhandler` work? It first queries the event that has been signaled, using `wd'q'`, and assigns the result to a global variable `wdq`:

```
wdq
+-----+-----+
|syshandler |mywin_handler |
+-----+-----+
|sysevent   |mywin_pressme_button |
+-----+-----+
|sysdefault |mywin_default |
+-----+-----+
|sysparent  |mywin |
+-----+-----+
|syschild   |pressme |
+-----+-----+
|systype    |button |
+-----+-----+
|syslocale  | |
+-----+-----+
```

syshwndp	1388	
+-----+	+-----+	+-----+
sysfocus	pressme	
+-----+	+-----+	+-----+
syslastfocus	pressme	
+-----+	+-----+	+-----+
sysinfo	1 552 146 200 200 192 173 800 600	
+-----+	+-----+	+-----+

Note that `wd'q'` only returns information about the last event that occurred. Re-running it will provide new information only if another event has occurred, otherwise it will give a domain error.

The result `wdq` is a boxed array describing the event and the current state of the form. The first column contain various identifiers, and the second column corresponding values. Note that the first three rows correspond to the three levels of event handler discussed above. `wdhandler` checks whether any of these event handlers exist, then

- defines each name in the first column with the corresponding value in the second column, for example a global variable `sysfocus` will be defined with the value `pressme`
- executes the first event handler it has found.

As another example, click on the form to give it focus, then press the Esc key. Click on the J session window, and look at the variable `wdq`:

```
wdq
+-----+-----+
| syshandler | mywin_handler |
+-----+-----+
| sysevent   | mywin_cancel  |
+-----+-----+
| sysdefault | mywin_default |
+-----+-----+
...
```

This shows that the second-level event handler for the Esc key is named

mywin\_cancel. Define a verb of this name to close the form:

```
mywin_cancel=: wd bind 'pclose'
```

Now click on the form to give it focus, press the Esc key, and the form will close.

## Entering Information

We next look at a simple example of creating a form to enter information. This example is in a script file included with the J system.

First, clear out any existing definitions with:

```
clear''
```

Then open the script file:

```
open 'system\examples\demo\name.ijs'
```

You may find it helpful to print out the script file; to do so, with the script window in focus, select the menu item File/Print. The relevant definitions are as follows:

```
NAME=: 'Jemima Puddle Duck'
```

```
EDITNAME=: 0 : 0
pc editname;
xywh 5 5 70 10;cc name edit;
xywh 85 5 32 12;cc OK button;
xywh 85 20 32 12;cc Cancel button;
pas 6 6;pcenter;pshow;ptop;
)
```

NB. this creates and initializes the form:

```
editname=: 3 : 0
wd EDITNAME
wd 'set name *',NAME
wd `''''pshow'
)
```

NB. this handles the Cancel button:

```
editname_Cancel_button=: wd bind 'pclose'
```

NB. this handles the OK button:

```
editname_OK_button=: 3 : 0
```

```
NAME=: name
wd 'pclose'
```

)

NB. run the form:  
`editname''`

The script defines a global variable, `NAME`, and a form, `EDITNAME`, with an edit field and OK and Cancel buttons. The following verbs are also defined:

- `editname` is used to create the form. It first resets the Window Driver, then sends the form definition `EDITNAME` to the Window Driver to create the form, then sets the value of the global `NAME` into the name field.
- `editname_Cancel_button` is used to handle a click on the Cancel button. It simply closes the form.
- `editname_OK_button` is used to handle a click on the OK button. It redefines the global `NAME` with the current value of the name field, then closes the form.

Try this out by running the script (select menu option Run/Window):

Change the value of the name field, press OK, and check the value of the global, `NAME`.

## Form Locales

A key point about forms is that they may be created and run in any locale, in fact this would typically be the case. Forms can be created as a class, then instantiated as an object when they are to be run. For a description, see the labs on *Locales* and *Object Oriented Programming*.

When a form is created, the current locale is recorded as the form locale. This locale is part of the event information, and allows an event to be handled by the form handler in the locale.

For example, this means a form can be run in its own locale, without conflicting in any way with definitions in other locales. You can design a form in the base locale, and run it without change in another locale.

To experiment with this, switch to the J session, and clear out existing definitions in the base locale:

```
clear''
```

Check there are no definitions in the base locale:

```
names''
```

Load the form into a locale myname:

```
'myname' load 'system\examples\demo\name.ijs'
```

The form is shown. Change the name and click OK to close the form and update the global, NAME. Note that there are still no definitions in the base locale:

```
names''
```

However, there *are* definitions in the myname locale:

```
names_myname_''
```

EDITNAME	NAME
editname	editname_Cancel_button
editname_OK_button	wdq

Read the value of the name defined:

NAME\_myname\_  
Squirrel Nutkin



## Other Message Handlers

`wdhandler` is the main message handler, but other handlers may be used if appropriate. There are essentially three alternatives:

- You can replace `wdhandler` with your own definition. This gives you complete control over the way you respond to events.

- You can define a specific handler function for a given form, using the `wd` command `phandler`. For example, when the following form is run, any event specific to it will invoke the sentence: `abc_event 0`:

```
wd 'pc abc;pshow;ptop'  
wd 'phandler "abc_event 0"  
abc_event=: wdinfo bind 'abc form handler'
```

- You can bypass the handler mechanism entirely by issuing the `wd` command `wait`, described below.

## Wait

The `wait` command allows you to create a form which bypasses the standard event handler mechanism. When it is used, other forms are disabled - this is referred to as a *modal* form, and is typically used where you want the user to respond to a query, or acknowledge an information message. The mechanism is similar to that used in Windows Common Dialog Boxes.

The `wait` command shows your form, disables events except for that form, and waits for an event. After an event occurs, you can use `wd 'q'` to query the event and respond to it. Events for other forms are only enabled when the form is closed.

For example, the following creates a form with a list box. When you select an entry, and click OK, the form is closed and the index of your entry is returned. While the form is running, other J windows are disabled.

```
PICKNUM=: 0 : 0
pc picknum;
xywh 70 9 34 12;cc ok button;cn "OK";
xywh 8 8 50 57;cc nos listbox;
pas 6 6;pcenter;
)

picknum=: 3 : 0
wd PICKNUM
wd 'set nos zero one two three four'
wd 'setselect nos 0'
wd 'wait'
res=. wd 'q'
wd 'pclose'
ndx=.({."1 res)i.<'nos_select'
".>{:ndx{res
)
```

When you use a wait command, take care that the form has some means of closing itself, for example, by defining the form with style `closeok`, or explicitly closing the form after the wait is ended.

## System Events

Some events are not attached to a specific form, and are considered as system events - these are the timer and DDE events. These events have a `sys_` prefix for their handler names, instead of the `formname_` prefix used for form events. For example, the `wd 'q'` result for a timer event is:

```
+-----+-----+
|syshandler|sys_handler|
+-----+-----+
|sysevent  |sys_timer   |
+-----+-----+
|sysdefault|sys_default|
+-----+-----+
```

The `wd 'q'` result for a `ddepoke` event is:

```
+-----+-----+
|syshandler|sys_handler|
+-----+-----+
|sysevent  |sys_ddepoke|
+-----+-----+
|sysdefault|sys_default|
+-----+-----+
```

To respond to such events, you define an appropriate event handler just as for a form event handler.

For example, define a handler for a timer event that writes the current time to the session, then set the timer to be 1000 milliseconds. The timestamps when Windows signals the timer event are written to the current session

```
sys_timer=: (6!:0) (1!:2) 2:
```

```
wd 'timer 1000'
```

```
1996 1 3 10 15 37.72
```

```
1996 1 3 10 15 38.76
```

```
1996 1 3 10 15 39.81
```

```
1996 1 3 10 15 40.9
```

To switch off the timer, create a new execution session (i.e. a jx window) and enter:

```
wd 'timer 0'
```

## **Window Controls**

[Overview p114](#)

[Parent Windows p115](#)

[Location and Size p116](#)

[Child Controls p117](#)

[Child Classes p118](#)

[Richedit Control p119](#)

[Statusbar p120](#)

[Tab Control p121](#)

[Toolbar p122](#)

[Common Dialog Boxes p123](#)

[Fonts p124](#)

[Accelerator Keys p125](#)

[Menus p126](#)

[Tab and Cursor Keys p127](#)

[Ownerdraw p128](#)

## Window Controls Overview

Windows supports several types of graphic controls that can be included in the user interface. All these controls are accessed using the Window Driver.

This chapter describes each control, plus other Window Driver commands used in developing user interfaces. For an example illustrating the main types of control, run menu Studio|Demos|controls

All controls are displayed in a window called the *parent* window. The parent window is created first, and controls are then added to it. The controls in a parent window are referred to as *child* controls.

The parent window and child controls have names, referred to as their *ids*, that are set when created. These start with an alphabetic character and consist of alphanumeric characters. Ids are case-sensitive and can be up to 31 characters. For example:

```
wd 'pc abc'
```

creates a parent window with id abc.

Several parent windows may be created at a time. Only one parent window may be *selected* at a time: `wd 'psel abc'` selects the parent with id abc. The `wd` commands affect the selected parent.

## Parent Windows

The parent window is the display area for the menus and child windows required by a user interface. The command `pc`, optionally followed by various styles, creates a new parent window:

- `pc` creates a standard parent window
- `pc owner` creates a parent window that is owned by the currently selected window. The owner is disabled until the new parent is closed.
- `pc dialog` creates a window that has a dialog box frame, that is, it cannot be resized.
- `pc closeok` creates a window that closes without an event when the user selects the window control menu item: Close.

Parent windows are initially not visible. They can be displayed in various ways using the `pshow` command, for example `pshow sw_showmaximized` shows the parent window maximized (to fit the available screen). Typically, a parent window and all its children are created first, then the `pshow` command is given.

## Location and Size

The location and size of controls in a parent is set with the *rectangle*. The `xywh` command sets all values for the rectangle.

4 integers define the rectangle:

x units from the left

y units from the top

w units in width (to the right)

h units in height (to the bottom)

These units are in terms of the average character size of a notional *system font*. The size of this system font can be obtained from the `qm` (query metrics) command. `wd 'qm'` returns the screen size in pixels, and the pixel size of the system font.

- x and w values are 1/4 of the system font width
- y and h values are 1/8 of the system font height

For example:

```
wd 'xywh 40 80 100 160'
```

sets the rectangle to be 40 units from the left edge of the parent; 80 units down from the top edge of the parent; 100 units wide; and 160 units high.

Since location and size are defined relative to the size of the notional system font, window definitions should display reasonably well on different screens at various resolutions.



## Child Controls

Child controls are created in the currently selected parent window. A parent window may have several child controls. `wd` commands affect the child controls of the selected parent window.

Child controls are created with a given class and styles. The class defines the type of control, the styles customize the control. Some styles are specific to a particular type of control, other styles can be used with most or all controls. Style names reflect their use, for example `es_uppercase` is a style applicable only to edit boxes. Styles whose names begin with the letters `ws_` apply to any control. The `group` style groups the control with the previous control. Tab and Cursor keys recognize groups.

The command `cc` creates child controls. Typical syntax is:

```
cc id class style1 style2 ...
```

## Child Classes

The class defines the appearance and behavior of the control. The classes are:

button	pushbutton
check box	check box
combobox	combination of edit and listbox
	dropdown combo box
combodrop	
combolist	combination of edit and listbox
edit	single line edit box
editm	multiline edit box
groupbox	group box
isigraph	graphics box
isipicture	picture from a BMP, WMF, or ICO file
listbox	list box
radiobutton	radio button
scrollbar	horizontal scrollbar
scrollbarv	vertical scrollbar
static	static text
staticbox	display box
progress	progress bar
richedit	single line rich edit control
richeditm	multiline rich edit control
spin	horizontal spin button
spin	vertical spin button
tab	tab control
trackbar	horizontal trackbar
trackbarv	vertical trackbar

***button***

A *button* simply initiates a Windows event when pushed. Possible styles are:

- `bs_defpushbutton`

the default pushbutton. Pressing Enter has the same effect as clicking this button.

- `bs_ownerdraw`

a button with a picture from a graphics file ***checkbox***

A *checkbox* allows a two way selection - checked or unchecked. The only style is:

- `bs_lefttext`

the caption is displayed on the left, instead of on the right. Use the `set` command to set the value of a checkbox, for example, to check control `cb`:

```
wd 'set cb 1'
```

The result of `wd 'q'` contains the current value of each check box.

***combobox***

***combodrop***

***combolist***

A *combobox* is a combination of a listbox and an edit box. You can enter a response into the edit box, or select a response from the listbox. You may select only one item in a combobox.

A *combodrop* is similar to a combobox, but initially shows only the edit box, and allows the list box to appear when selected.

A *combolist* is similar to a combobox, except that you may select only from the list box, and may not enter any response into the edit box.

Class styles are:

- `cbs_autohscroll`

allow horizontal scrolling of the edit box

- `cbs_sort`

sort entries alphabetically Use the `set` command to set the possible selections in a checkbox. This can be a list of names, that may be delimited by `LF`, `EAV` or `"`. For example:

```
wd 'set cbox red green blue brown'
wd 'set cbox "red" "green" "blue" "brown"'
wd 'set cbox ',; (::'red green blue brown') ,each LF
```

Use `setselect` to set the selection. For example, the following sets item 2:

```
wd 'setselect cbox 2'
```

The result of `wd 'q'` contains the current value and selection of each combobox. For example:

```
...
+-----+-----+
| cbox          | blue          |
+-----+-----+
| cbox_select   | 2             |
+-----+-----+
```

## *edit*

### *editm*

An *edit* control is a single-line box, and an *editm* control is a multiline box, in which text can be displayed and edited. Edit control styles are:

- `es_autoscroll`

use horizontal scroll bars if required

- `es_autovscroll`

use vertical scroll bars if required

- `es_center`

center text

- `es_lowercase`

lowercase text only

- `es_readonly`

display text as read-only

- `es_right`

right justify text in control

- `es_uppercase`

uppercase text only Use the `set` command to write text to an edit control. For example:

```
wd 'set edit *Jeremy Fisher'
```

The result of `wd 'q'` contains the current text for each edit box, and indices of any selected text:

```
...
+-----+-----+
|edit      |Peter Rabbit|
```

```
+-----+-----+
|edit_select |6 12|
+-----+-----+
```

### ***groupbox***

A groupbox control is a box used to group controls, most often radiobuttons. There are no class styles.

### ***isigraph***

An isigraph control is a window that can display graphics. First create an isigraph control, then use graphics commands to create and display the graphics. All graphics commands start with `g`. Graphics are displayed only when the `gshow` command is given.

Graphics commands apply to the currently selected isigraph control. The graphics window has coordinates running from (0,0) in the bottom left hand corner to (1000,1000) at the top right.

See the demos, Studio\Demos\isigraph.

### ***isipicture***

An isipicture control displays an image from a file. The file may be a DIB (device independent bitmap), WMF (Windows metafile) or ICO (icon file). DIB files usually have a file extension of `.bmp`.

The `set` command sets the filename. For example, to display the J logo icon:

```
wd 'set isipicture *system\examples\data\j.ico'
```

### ***listbox***

A listbox control displays a list of items that can be selected by the user. Listbox control styles are:

- `lbs_extendsel`

extended select holding down the Shift key

- `lbs_multicolumn`

display in multiple columns

- `lbs_multipleselel`

allow multiple selections

- `lbs_ownerdrawfixed`

create an ownerdrawn listbox

- `lbs_sort`

sort entries alphabetically Use the `set` command to put items in the listbox (as in combobox above).

In the result of `wd 'q'`, multiple selections are delimited by `LF`. For example:

```
+-----+-----+
|listbox      |green blue brown      |
+-----+-----+
|listbox_select|1 2 3                  |
+-----+-----+
```

### ***progress***

A progress control displays a bar used to indicate the status of a program. Use the `set` command to set values between 0 and 100. For example, to indicate the half way stage:

```
wd 'set pg 50'
```

### ***radiobutton***

`radiobutton` controls typically allow a single selection from a group of controls. Selecting one control switches the others off. The only style is:

- `bs_lefttext`

the caption is displayed on the left, instead of on the right. Use the `set` command to set the value of a radiobutton, for example, to check control `rb`:

```
wd 'set rb 1'
```

The result of `wd 'q'` contains the current value of each radiobutton.

### ***richedit***

#### ***richeditm***

A *richedit* control is a single line edit box, and a *richeditm* control is a multiline edit box, each containing rich edit text. Class styles are:

- `es_ahscroll`

use horizontal scroll bars if required

- `es_ahscroll`

use vertical scroll bars if required

- `es_center`

center text

- `es_readonly`

display text as read-only

- `es_right`



right justify text in control

- `es_sunken`

display `sunken` For more information, see the `Richedit Control` section later in this chapter.

### ***scrollbar***

#### ***scrollbarv***

A *scrollbar* control displays a horizontal scrollbar, and a *scrollbarv* displays a vertical scrollbar. There are no class styles.

The `set` command sets four values for the scrollbar: leftmost (or top) position, current position, rightmost (or bottom) position, and the size of change resulting from a click in the area between the thumbbox and either end of the scrollbar.

```
wd 'set sb 0 500 1000 50'
```

A single parameter sets the current position:

```
wd 'set sb 600'
```

Clicking a scrollbar signals an event. The result of `wd 'q'` has the current position of the scrollbar. For example:

```
...
+-----+-----+
|sb      |550      |
+-----+-----+
```

### ***spin***

#### ***spinv***

A *spin* control displays two arrows horizontally, and a *spinv* control displays the arrows vertically. Clicking an arrow initiates a Windows event, and the result of `wd 'q'` is `_1` for the down or left arrow, `1` for the up or right arrow.

### ***static***

A static control is used to display text. It is never active. Static control styles are:

- `ss_center`

center text

- `ss_leftnowordwrap`

left justify, no word wrap

- `ss_noprefix`

allow & characters in control

- `ss_right`

right justify text

- `ss_simple`

static text control ***staticbox***

A static control is used to display boxes with various types of frames and backgrounds. It is never active. Staticbox control styles are:

- `ss_blackframe`

black frame

- `ss_blackrect`

black rectangle

- `ss_etchedframe`

etched frame

- `ss_etchedhorz`

etched horizontally only

- `ss_etchedvert`

etched vertically only

- `ss_grayframe`

gray frame

- `ss_grayrect`

gray rectangle

- `ss_sunken`

sunken

- `ss_whiteframe`

white frame

- `ss_whiterect`

white rectangle

***tab***  
A tab control is used to display several Windows controls on tab forms. Class styles are:

- `tcs_button`

display tabs as buttons

- `tcs_multiline`

allow tabs to be displayed in several lines For more information, see the Tab Control section later in this chapter.

### ***trackbar***

#### ***trackbarv***

A *trackbar* control displays a horizontal trackbar, and a *trackbarv* displays a vertical trackbar. Class styles are:

- `tbs_autoticks`

display tick marks along control

- `tbs_both`

display tick marks on both sides of control

- `tbs_enablese xrange`

enables `setselect` to mark a range on the trackbar

- `tbs_left`

display thumb pointing to left / tick marks on left

- `tbs_nothumb`

do not display a thumb control

- `tbs_noticks`

do not display any tick marks

- `tbs_top`

display thumb pointing to top / tick marks on top The `set` command sets up to five values for the trackbar: leftmost (or top) position, current position, rightmost (or bottom) position, and the size of change resulting from a click in the area between the thumbbox and either end of the trackbar, and line size.

```
wd 'set tb 0 3 20 1 1'
```

A single parameter sets the current position:

```
wd 'set tb 4'
```

Clicking a trackbar signals an event. The result of `wd 'q'` has the current position of the trackbar. For example:

```
...
+-----+-----+
|tb           |4           |
+-----+-----+
```

## Richedit Control

A richedit control is an edit control that displays RTF (rich text format) data.

RTF is a text description language that uses only standard ASCII characters, so that you can create and view RTF data as ordinary text. For a summary of the language, see file: system\examples\data\rtf.txt.

You can also create RTF data from most Windows editors, for example, WordPad, by saving your text in RTF format. Thus you could create and format some text in WordPad, save it in RTF format, then read in the RTF file and display it in J using a richedit control.

You can also copy and paste between a richedit control and any application, such as WordPad, that supports RTF.

Events and event data for richedit controls are the same as for edit controls. The event data is the simple text from the control, not the rtf data. To read the text in rtf format, use command `qrtf`.

Here is a summary of commands applicable to richedit controls:

- `set`

set rtfdata into a richedit control. For example:

```
wd 'set rid *',rtfdata
```

- `setreplace`

replace selected rtfdata in a richedit control For example:

```
wd 'setreplace rid *',rtfdata
```

- `setbkgnd`

set background color. For example:

```
wd 'setbkgnd rid 255 0 0'
```

A useful color is the gray window background, which is typically 192 192 192. A richedit control with this background color and without the sunken style appears as text on the form.

- `setreadonly`

prevent the user from making changes. For example:

```
wd 'setreadonly rid'
```

- `qrtf`

reads the RTF data from the control. For example:

```
rtfdata=. wd 'qrtf rid'
```

Here is a brief overview of the RTF format:

The `\` character starts an RTF command and curly braces `{ }` group data.

Some RTF commands:

`\b`

**bold**

`\cfn`

color from color table

`\fn`

font from font table

`\fsn`

font size

`\i`

italic

`\par`

paragraph (new line)

`\ul`

underline Commands like `\b` can be followed by a 0 or 1 to turn the attribute off or on.

The font table definition is enclosed in `{ }` and each font definition is also enclosed in `{ }`. For example:

```
{\fonttbl{\f0\fcourier Courier New;}}
```

The `\fn` command indicates which font to use. For example:

```
\f0 test
```

The `\fsn` command indicates font size. For example:

```
\fs90 test
```



To replace the current selection with "test" in Courier New size 90:

```
wd 'setreplace rid *{\fonttbl{\f0\fcourier Courier New;}}\f0\fs90 test}'
```

Colors are managed in a manner similar to fonts. Define a color table and then select colors from that table. The following table defines colors black and red:

```
{\colortbl\red0\green0\blue0;\red255\green0\blue0;}
```

To replace the current selection with 'test' in red:

```
wd 'setreplace rid *{\colortbl\red0\green0\blue0;\red255\green0\blue0;}\cf1 test}'
```

The / character is an escape character that treats the following character as text. For example to enter a } as part of text, rather than treat it as a grouping character:

```
wd 'setreplace red *{the character /{ appears}{'
```

The following example (in script examples\demo\rtf.ijs) creates some RTF data, then displays it in a form with a richeditm control:

```
rtfdata=: 0 : 0
{
{\fonttbl{\f0\fcourier Courier New;}}
{\colortbl\red0\green0\blue0;\red255\green0\blue0;}
\f0 black
\par
\b
\cf1 bold red
\par
\i
\fs60 big bold italic red
}
)
wd 0 : 0

pc abc closeok;
xywh 148 8 34 12;cc ok button;cn "OK";
```

```
xywh 148 23 34 12;cc cancel button;cn "Cancel";
xywh 9 7 128 69;cc rid richeditm es_autovscroll es_sunken;
pas 6 6;pcenter;
rem form end;
)
wd 'set rid *',rtfdata
wd 'pshow'
```

## Statusbar

A statusbar can be shown at the foot of a form.

There are 3 statusbar commands:

```
sbar n;                n is number of panes in the statusbar  
sbarshow b;           b is 0 to hide, 1 to show (default)  
sbarset id width text; a width of -1 leaves at current value
```

The first pane is special. Its width is variable and takes up what is left. It does not have a border. It can be used to display help text for menu and toolbar items.

## Tab Control

A tab control is a panel used to display other controls.

Commands applicable to the tab control are:

<code>set tab text;</code>	add a tab label to the control
<code>setreplace tab n text;</code>	replace the text of a tab label
<code>setinsert tab n text;</code>	insert a tab label
<code>setdelete tab n;</code>	delete a tab label

There are two uses for tab controls:

- You can set up a single set of controls, and use the tab control to switch between different values for the controls. In this case, the controls in the tab area are created in the same form as the tab itself.
- You can set up several forms, and use the tab control to switch between forms. In this case, the forms should be created separately from the main form.

As an example of the first use, you could create controls that contain values for days of the week. Selecting the tab for the day changes the values of the controls accordingly.

For example, the following demo shows an edit control that depends on the day of week:

```
load 'system\examples\demo\days.ijs'
```

An example of the second type of tab control is the controls demo that displays the main Windows controls supported by J, see menu Studio|Demos|Controls.

In this example, the controls displayed on each tab were created in separate script files: `system\examples\demo\control1.ijs`, `control2.ijs` etc.

This second type of tab control uses the `creategroup` command to allow several forms to be displayed in a single form.

A form definition used with a `creategroup` command is an ordinary form that can be designed and tested with the form editor. It is loaded by the parent of the tab control by doing a `form_run` that is bracketed by `creategroup` commands. The first `creategroup` command gives the id of the tab control where the new controls are being created. The final `creategroup` command has no parameter.

`creategroup` causes parent commands such as `pc` to be ignored so that when the form definition is run, the child creates occur in the original parent form. The initial argument to `creategroup` is an id, usually of a tab control, in the current form. Controls created under a `creategroup` command are created as hidden, and as part of a group with the id from their (ignored) `pc` command. The `setshow` command with a group id, shows or hides the controls in a group.

For example, here is the relevant code from the controls demo:

```
wd 'creategroup tabs'  
edits_run''  
selects_run''  
wd 'creategroup'
```

This code:

- uses `creategroup` to prepare to load forms for the `tabs` control
- loads each form to be shown on the `tabs` control
- uses `creategroup` with no parameter to finish up

Note that a tab control should be created before any controls that appear on top of it – this ensures controls will be painted properly.

The event data for a tab control is the label text and the `id_select` variable contains the index of the selected tab.

## Toolbar

A toolbar can be shown at the top of a form, beneath any menu.

There are 3 toolbar commands:

```
tbar filename;
```

filename of toolbar bitmap

```
tbarshow b;
```

b is 0 to hide, 1 to show (default)

```
tbarset id index image;
```

toolbar index and image number in bitmap The `tbarset` command with an empty id sets the toolbar index as a separator with a width as specified in the image value.

Toolbar buttons are usually for a command that is also on the menu. If the menu item and the toolbar button are given the same id, then `set` and `setenable` commands affect both, and they will cause the same event.

The `set` command for an id that is a menu or toolbar command will check or uncheck the menu or toolbar. The `setenable` command for an id that is a menu or toolbar command enables or disable the commands.

Any toolbar bitmap file can be used. An example is provided in file: `system\examples\data\isitbar.bmp`, which is referenced by the controls demo.

You can create and edit toolbars using Paint. To do so, open the .bmp file using Paint, maximize the zoom setting and select grid on.

## Common Dialog Boxes

Windows provides several built-in dialog boxes called *Common Dialog Boxes* that perform useful functions.

The *color* dialog box allows selection of colors. The result is a list of the RGB values for 17 colors, of which the first is chosen in the standard color dialog box, and the rest in the custom color dialog box. Try:

```
_3[\ ". wd 'mbcolor'
```

This returns a matrix of the 17 values, one row per color.

The *font* dialog box allows selection of a logical font. The result describes the font chosen (in this case the font used in the J logo):

```
wd 'mbfont'
"Bookman Old Style" 24
```

The *message box* dialog box displays a message and waits for a user response. For example:

```
wd 'mb title text'
wd 'mb title text mb_iconstop mb_retrycancel'
```

To list all message box styles:

```
list wd 'qs mb'
mb_abortretryignore mb_defbutton2      mb_defbutton3
mb_iconasterisk      mb_iconexclamation mb_iconhand
mb_iconinformation   mb_iconquestion    mb_iconstop
mb_ok                 mb_okcancel        mb_retrycancel
mb_yesno              mb_yesnocancel
```

The *open filename* dialog box allows the user to select a fully qualified file name:

```
wd 'mbopen'
```

```
wd 'mbopen title "" "" Write(*.wri)|*.wri| Word(*.doc) |*.doc"
ofn_filemustexist'
```

To list all open filename styles:

```
list wd 'qs mbopen'
ofn_createprompt ofn_filemustexist...
ofn_overwriteprompt ofn_pathmustexist
```

The *save filename* dialog box is similar to the open filename dialog box:

```
wd 'mbsave'
```



## Fonts

A font can be chosen with the `mbfont` command. The result is a description of a font, that can be used as the argument to other commands.

A maximum of 20 fonts can be selected for use in controls. Deleting a control does not release the font resource. A `wd 'reset'` command frees all font resources.

Use the `setfont` command to set the font. For example, the following sets the font for control `bn` to: Lucida Console, 24 point, italic, bold:

```
wd 'setfont bn "Lucida Console" 24 italic bold'
```

## **Accelerator Keys**

& in the name of a button or menu item sets a keyboard accelerator. The & is not displayed and the next character is underlined. Pressing ALT + the character is the same as clicking on the button or menu item.

## Menus

Several Window Driver commands support menus:

```
menu id text
```

```
add menu item
```

```
menupop text
```

```
add popup menu item
```

```
menupopz
```

ends popup menu and drops down a level

```
menusep
```

add separator line in a popup menu To add a popup menu item `pop1`, displayed in the menu bar:

```
wd 'menupop pop1'
```

To add individual items to the popup menu, displayed when the menu is selected:

```
wd 'menu item1 "item name"'
```

This displays the text "item name", and if selected, the Windows result contains the name `item1`.

To add a separator line:

```
wd 'menusep'
```

To end the popup menu:

```
wd 'menupopz'
```

To check a menu item:

```
wd 'set item 1'
```

NB. 1=check, 0=uncheck

To enable a menu item:

```
wd 'setenable item 1'
```

NB. 1=enable, 1=disable

## Tab and Cursor Keys

TAB and SHIFT+TAB keys cycle the focus through the children in the order they were created.

Cursor keys cycle through the controls in a *group*. By default, controls, except for radiobuttons, are created as part of a group consisting only of themselves.

## Ownerdraw

A button created with the *bs\_ownerdraw* style is an ownerdraw button. A listbox created with *lbs\_ownerdrawfixed* style is an ownerdraw listbox.

A control with ownerdraw style displays a picture from a file. The files supported are the same types supported for the pictures in isipicture class windows.

The file names to display are set with the `set` command.

For example, the following will display an ownerdraw button with the J icon:

```
wd 'pc abc; xywh 10 10 30 30'  
wd 'cc b1 button bs_ownerdraw'  
wd 'cn "system\examples\data\jb.ico" '  
wd 'pas 10 10; pshow'
```

## Window Driver Command Reference Overview

The foreign family 11! :x is the J-Windows interface.

All 11! :x verbs are rank 1.

[wd commands p130](#) are executed when x is 0 ( 11! :0 ).

[gl2 commands p131](#) are for 2D (GDI) graphics (x between 2000 and 2999).

[fontspec p132](#) is several parameters that specify a font.

[isigraph events p133](#) support character and mouse events.

[Mapping Mode p134](#) affects how logical units are mapped onto the display surface.

[gl3 commands p135](#) are for 3D (OpenGL) graphics (x between 3000 and 3999)

[OpenGL printing p136](#) for printing OpenGL images by creating a bitmap RC, drawing to the bitmap, and then printing the bitmap.

## **wd commands 11!0**

wd takes a string right argument and returns a string result. A wd argument is 0 or more statements delimited by semicolons. A statement is a command followed by 0 or more parameters. Commands and parameters are separated by one or more whitespace characters from the set: space, carriage return, line feed, and tab. Simple parameters start with an alphanumeric, -, or \_, and run to a; or whitespace. A \* starts a parameter at the next character, and runs to the end of the array. Delimited parameters start with " or 255{a. (EAV) and run to the matching delimiter. Some commands are not supported in some environments.

A wd error is a J domain error. Command qer returns the error message text and wd argument index for the last error.

The first letters of a command usually indicate a category: dde, mb (message box and common dialogs), menu, ole, p (parent), q (queries), set (setting properties), sm (session manager), and vbx.

A parameter can be an id, style, number, or text. An id identifies a parent, child, menu item, or other object. A style is a keyword, usually with a prefix. For example, bs\_autoradiobutton is a button style.

A color is 3 parameters giving RGB values in the range 0 to 255. For example, 0 0 0 is black, 255 255 255 is white, and 255 0 0 is red.

A bool is 0 or 1 and is 1 if elided.

## **wd Commands**

**beep [i] ;** beep (parameter is ignored)

**cc id class [style...]** ; create child

**clipcopy text ;** put text on the clipboard

**clippaste ;** result is text from the clipboard



**cn name ;** set child name after a cc command (otherwise setcaption is preferred)

**creategroup id ;** causes parent commands to be ignored so that, when a form definition is run, the child controls created are in the current form. The id is a control in the current form, usually a tab control, and child creates are offset from that control. Controls created under creategroup are hidden and are part of a group with the id from their ignored pc command. The setshow command with a group id hides or shows the controls in a group. A form definition loaded under creategroup is a normal form definition that can be designed and tested with the form editor. It is loaded by a main form with a form\_run that is bracketed by creategroup commands. The first creategroup gives the id of a control in the main form and the final creategroup command has no parameter and ends the group.

**creategroup ;** ends the group started by a previous creategroup command

**dde commands...**

**fontdef [fontspec p132](#);** default font used when a child is created.

**mb title text [style...]** ; messagebox with styles from set (mb\_arbortretryignore, mb\_defbutton2, mb\_defbutton3, mb\_iconasterisk, mb\_iconexclamation, mb\_iconhand, mb\_iconinformation, mb\_iconquestion, mb\_iconstop, mb\_ok, mb\_okcancel, mb\_retrycancel, mb\_yesno, mb\_yesnocancel)

**mbcolor [colors]** ; choose color common dialog box. Result is chosen color and 16 custom colors. Colors are 3 integers giving RGB values. Optional argument sets initial chosen color and 16 custom colors (51 integer values in range 0 to 255).

**mbfont [fontspec p132](#)** ; choose font common dialog. Start with last chosen and set new one if OK. Result is a [fontspec p132](#).

**mbopen title directory filename filterpairs [style...]** ; open file common dialog returns fully qualified file name selected by user. Filterpairs are delimited by |. Styles are from the set (ofn\_createprompt, ofn\_filemustexist, ofn\_nochangedir, ofn\_overwriteprompt, ofn\_pathmustexist).

```
wd 'mbopen'
wd
'mbopen mytitle "" "" "Write(*.wri)|*.wri|(Word(*.doc)|*.doc"
ofn_filemustexist'
```

**mbprinter [pd\_printsetup]** ; The parameter selects the setup dialog instead of the print dialog. The result is " if the user pressed CANCEL. If the user pressed OK, the result is the name of the selected printer followed by LF delimited values. Currently the only value after the name is the print-to-file value from the print dialog. This sets the session default printer setup that is used by glzcreate with no argument.

**mbsave title directory filename filterpairs [style...]** ; save file common dialog returns fully qualified file name (see mbopen).

**menu id text** ; add menu item

**menupop text**; add popup menu item

**menupopz** ; ends popup menu and drops down a level

**menusep** ; separator line in a popup menu

**ole commands...**

**pactive** ; SetActiveWindow for selected parent

**pas i j** ; parent size adjusted to provide i and j margins beyond children

**pc id [style...]** ; parent create. Styles are: nomenu, nomin, nomax, nosize, dialog, owner, and closeok.

**pcenter** ; center parent on screen

**pclose** ; close parent. If no parent selected, nothing is done and there is no error.

**pcolor R G B**; (win32) set parent background color. See setcolor command.

**pgroup groupname**; set parent groupname. Parents are created with a default groupname of ". The ijsx window has a groupname of 'ijx' and an ijs window has a groupname of 'ijs'. A groupname can be used as an argument to the reset and qp commands.

**picon filename n** ; (win32) set form icon with icon n from file (exe, dll, or ico file)

```
wd'picon "',(jsystempath  
'system\examples\data\jy.ico'),'" 0'
```

**pmove x y w h ;** move and resize parent. Values are in logical units and are relative to the top left corner of the screen. Value of -1 inherits current value.

**pmovex x y w h ;** move and size form in pixels

**pn text ;** name for parent window caption

**psel id ;** select parent id to be target for subsequent commands

**psel n ;** select parent with qhwnp of n for subsequent command

**psel ;** clear parent and child selection

**pshow [style] ;** style is from the set: sw\_hide, sw\_minimize, sw\_restore, sw\_show, sw\_showmaximized, sw\_showminimized, sw\_showminnoactive, sw\_showna, sw\_shownoactivate, sw\_shownormal. sw\_shownormal is the default.

**ptop [bool];** set 1 so that window stays on top of other windows. No error on Jwdp (portable Java version), but has no effect.

**q ;** return event data

**qbreak ;** (win32) returns count of menu/button/and toolbar commands that have been ignored because J was busy. This value can be polled in a loop with a long execution time to see if the user is getting impatient.

**qchildxywh id;** return child position and size in units

**qchildxywhx id;** return child position and size in pixels

**qcolor color ;** return RGB for system colors defined in system\packages\color\wdcolor.ijs

**qd ;** return data for form. Similar to q result.

**qer ;** return last error information

**qformx ;** returns form x y w h values in pixels.

**qhinst** ; (win32) return HINSTANCE of application

**qhwndc id**; return HWND (handle) of child

**qhwndp** ; return HWND (handle) of current parent

**qhwndx** ; (win32) return HWND (handle) of application

**qkeystate keys** ; (win32) returns 0 or 1 (pressed) for each key in keys. The values in keys are virtual key values as defined in packages\graphics\vkeys.ijs.

The portable way to work with shift type info in an event is to use the sysmodifiers event data. The sysmodifiers value is: 2# .ALT , META , CTRL , SHIFT

**qm** ; return system metrics:

- screen width, screen height,
- x logical unit, y logical unit,
- cxborder, cyborder,
- cxfixedframe, cyfixedframe,
- cxframe, cyframe,
- cycaption, cymenu,
- desktop x, desktop y,
- desktop w, desktop h

More elements may be added.

**qp [groupname]** ; return parent ids. With no parameter it reports all windows with the default groupname of ". With a groupname parameter it reports all windows with that groupname.

**qprinters** ; (win32) returns an LF delimited list of printers. The first entry is the session default printer. The second entry is the system default printer. The remaining entries are a list of all printers. The session default printer starts out as the system default printer and is changed by mbprinter or File|Print Setup.

**qpx** ; extended information about all forms (see wdforms\_j\_)

**qrtf id**; (win32) return RTF data from richedit control

**qs [cmd]** ; return styles or keywords used in command. cmd can be: cc, gpen, mb,

mbopen, mbsave, pshow, sendkeys, smshow, or winexec

**qscreen** ; screen information (see glqprinter)

horzsize	width in millimeters
vertsize	height in millimeters
horzres	width in pixels
Vertres	height in raster lines (pixels)
logpixelsx	horizontal pixels per logical inch
logpixelsy	vertical pixels per logical inch
bitspixel	number of color bits per pixel.
planes	number of color planes
numcolors	number of entries in the device's color table (_1 if > 8)
aspectx	relative width of a pixel
aspecty	relative height of a pixel
aspectxy	diagonal width of a pixel

**qwd** ; returns 'jjava' for Jwdp and 'jwin32' for Jwdw

**rem text** ; remark

**reset [groupname]** ; close windows. With no parameter it closes all windows with the default groupname of ". With a groupname parameter it closes only windows that have that groupname (as set by the pgroup command).

**sbar n**; create status bar with n panes

**sbarset id width text**; set id, width, and text of the next statusbar pane. A width of -1 leaves the width unchanged. A set command sets the text of the pane.

**sbarshow [bool]**; show (1) or hide statusbar

**security [bool]**; (win32) 1 sets java sandbox (disables commands that can 'cause damage')

**set id [parameters...] ; set object property**

*checkbox, radiobutton* - check (1 or elided) or uncheck (0)

*combobox* - similar to listbox

*edit, editm* - set text.

*isipicture* - set filename of picture source. dib or bmp (device independent bitmap), wmf (windows metafile), or ico (icon) file to display.

*listbox* - clear contents and add items. A parameter can contain LF delimited items. Ownerdrawfixed listbox items are picture file names.

```
wd 'set listbox red green blue'
wd
'set listbox red green blue "pink",LF,'orange''
```

*menu* - check (1 or elided) or uncheck (0). If menu and toolbar button have same id, both are set.

*progressbar* - set value in range 0 to 100

*richedit, richeditm* - set RTF text

*scrollbar* - set min/position/max/pagesize values or just position

```
wd'set scrollbar 0 60 100 10'
wd'set scrollbar 75'
```

*static* - set text

*statusbar* - set pane text

*tab* - add a tab for each parameterwd'set tab Jan Feb Mar Apr'

*toolbar* - check (1 or elided) or uncheck (0). If menu and toolbar button have same id, both are set.

*trackbar* - similar to scrollbar but has linesize in addition to pagesize

```
wd'set trackbar 0 60 100 10 2'
```

**setbkgnd id color ;** (win32) set background RGB color of richedit or richeditm control

**setcaption id text ;** set control caption. If control is ownerdraw button, text is picture filename.

**setcolor id textR G B textbkgndR G B bkgndR G B ;** (win32) set control colors. You can set form and control (text, text background, and background) colors. The setcolor command can override the gray readonly edit boxes. The setcolor command has no effect on push buttons or the dropdown listbox of a combobox (unfortunate Window facts). For example:

```
abc_run=: 3 : 0
wd ABC
NB. initialize form here
wd 'pcolor 0 0 255'
wd 'setcolor ccreditm 255 0 0 0 0 255 0 0 255'
wd 'pshow;'
)
```

**setcolwidth id width ;** (win32) set multicolumn listbox column width

**setdelete id n ;** delete tab item n

**setedit id x ;** parameter is z x c v or y and it performs the command undo, cut, copy, paste, or redo

**setenable id [bool] ;** enable (1) or disable. Applies to menu items, toolbar buttons, and controls.

**setfocus id ;** set focus on control

**setfont id [fontspec p132](#);** set font for control

**setinsert id n text ;** insert tab item at position n

**setinvalid id ;** (win32) InvalidateRect

**setlimit id n ;** (win32) limit length of user typed text in edit, editm, richedit,

richeditm, combodrop, and combobox controls.

**setreadonly id [bool] ;** set readonly (1) for edit, editm, richedit, and richeditm

**setreplace id newtext ;** replace selected text with newtext in edit, editm, richedit, and richeditm control

**setreplace id n newtext ;** replace tab item n text with newtext

**setscroll id n ;** scroll n to be first visible line in editm and richeditm control

**setselect id [parameters...] ;** set selection

*edit, editm, richedit, richeditm* - parameters are [ start end [noscroll ] ]. Selects text from start to end. If start and end are elided, then all text is selected. If noscroll is elided or 1, then the selection is not scrolled into view. If noscroll is 0, then the selection is scrolled into view

*combobox, listbox* - -number of item to select. -1 clears selection.

*trackbar* - parameters are start and end of range to mark as selected

**setshow id [bool] ;** show (1) or hide. If it is a group id (the id of the pc command ignored by a creategroup), then all controls in the group are affected. Controls explicitly hidden are not affected by a group show.

**settabstops id [ n [ n [ n...] ] ] ;** sets tab stops in editm and listbox controls. Values are in dialog units.

**setupdate id ;** (win32) UpdateWindow

**setxywh id x y w h ;** set child position and size in units

**setxywhx id x y w h ;** set child position and size in pixels

**smcolor n R G B ;** set code editor color. No effect in Jwdp.

**sminputlog ;** return input log



**smkeywords n keywords** ; code editor keywords (for color n). No effect in Jwdp.

**smsetlog** ; set input log

**tbar filename**; set filename of bmp file for toolbar. JFEwdp (JFE wd portable) can read either bmp or gif files. The file contains images that are 16 pixels wide by 15 pixels high.

**tbarset id index image**; set id for a toolbar button. index is the position in the displayed toolbar and image is the index of the button bitmap to use from the toolbar bitmap.

**tbarshow [bool]**; show (1) or hide toolbar

**timer i** ; set interval timer to i milliseconds. Event systimer occurs when time has elapsed. The timer keeps triggering events until the timer is turned off. An argument of 0 turns the timer off. The systimer event may be delayed if J is busy, and it is possible for several delayed events to be reported as a single event.

**tnomsgs** ; (win32)

**wait** ; (win32) the normal message loop executes a handler in immediate execution when an event occurs. This is overkill for getting input from a simple form, and the way mb works is simpler: the message box is displayed, other events are disabled until it is closed, and the result is the wd result. The wait command allows a form to be used in a similar way. The wait command shows the form, disables events except for that form, and waits for an event. The event information is given by wd 'q'. While a wait form is active events in other forms, DDE events, and timer events are disabled. They are enabled when the wait form is closed.

**winexec text [style]**; Execute program. style is same as for pshow.

```
wd ' winexec "write.exe text.wri" sw_showmaximized;'
```

**xywh x y w h** ; sets rectangle. -1 value inherits the previous value.

## **dde commands (win32)**

**ddeadvice t i d** ; send data to client in an advise loop. Client will get ddeadvice

event.

**ddecons** ; return s|t active conversations

**ddedis** [s [t]] ; discontinue conversations

wd 'ddedis ;'	conversations with all servers
wd 'ddedis s ;'	conversations with servers
wd 'ddedis s t ;'	conversation with server s on topic t

**ddeex s t d** ; data d is sent to server s for topic t to execute

**ddename id** ; set dde service name (can be done only once). Command line /ddename= can also set dde service name.

**ddepoke s t i d** ; send data to S|T!I

**ddereq s t i** ; data for item i is requested from S|T

**ddereqd d** ; send data in response to ddereq event (must be done immediately after ddereq event)

**ddestart s t i** ; advise loop requested for S|T!I. New data signals ddeadvise event.

**ddestop s t i** ; stop advise loop on S|T!I

## **ole commands (win32)**

**oledlg id** ; run property dialog. State can be saved with the olesave command.

**oleenable id eventname [bool]** ; enable/disable event. You must enable an event in order to trigger an event in J.

**oleget id objectname property** ; return property value. Objectname is *base*, *temp*, or a name set with oleid. If the result is an object, it is set as the temp object. This allows a series of wd commands that use the temp object to get the next object.

**olegetlic** ; returns licence information

**oleid** ; give a temp object a name to make it permanent

**oleinfo id** ; return information about events, methods, properties, and constants

**oleload id filename** ; initialize properties from a file created by olesave. An oleload should only be done once before it is shown.

**olemethod id objectname method parameters....**; run a method. , is an elided parameter. A wd parameter of , is the same as "", except it is treated as an elided parameter where appropriate. Some methods distinguish between a numeric parameter and a string. A simple (not delimited) string that is an integer is treated as an integer. If you want 23 to be treated as a string, use "23". If the result is an object, it is set as the temp object.

An *object* parameter is indicated by a simple parameter of the form:object:formid.childid.objectname

A *picture object* parameter is indicated by a simple parameter of the form:picture:filename

**olemethodx** ;

**oleocx** ;

**olerelease** ;

**olesave id filename** ; save properties in a file that can be used to initialize a control after it is created

**oleset id objectname property value** ; set property value

**olesetlic** ; set the licence information for a runtime OCX

## **Incompatible changes**

197{a. is no longer supported as a parameter delimiter. Event data no longer includes sysinfo. Many commands have been decommitted.

## gl2 commands 11!:2000+n

gl2 commands are for 2D drawing in an isigraph control. Script system\main\gl2.js defines gl2 verbs in the jgl2 locale.

```
load 'gl2' NB. define gl2 verbs in jgl2 locale
```

The many gl2 definitions are in the jgl2 locale so they don't clutter up the z locale. Production users of gl2 can either use the full name (e.g. glline\_jgl2\_) or can use coinsert to add the jgl2 locale to their locale path. Requiring \_jgl2\_ on all names for casual use is a nuisance and you can use coinsert to add jgl2 to your. For example:

```
coinsert 'jgl2'
```

Most gl2 commands add drawing information to a buffer. When the control is painted, it is painted with these buffered commands. Some commands, such as glcursor are not buffered commands and either have an immediate effect, or change a global state.

[mapping mode p134](#) affects how the logical units of the gl2 commands are mapped onto the display surface.

[isigraph events p133](#) allow user interaction with the isigraph control with the keyboard and mouse.

Command 11!:2999 takes a list of multiple gl commands. Each command starts with an integer count followed by the command and data. For example:

```
11!:2999 [ 4 2013 500 500 4 2013 900 100 2 2036 NB. glline, glline, glshow
```

```
11!:2999 [ 4 2056 500 200 5 2038 65 66 67 2 2036 NB. gltextxy, gltext, glshow
```

This can be used to move the overhead in the J Engine Protocol of passing thousands of individual small commands over the socket interface by one large 2999 command.

---

# gl2 Commands

**glarc x y w h xa ya xz yz ;** (not wince) draw arc on the ellipse defined by rectangle. Arc starts at xa,ya and ends at xz,yz. Start and end points need not lie on the ellipse, they define a line from the center that intersects the ellipse.

**glbkcolor " ;** set current background color to current color (last grgb). Background color is used in the gaps in styled lines and as the background for text.

**glbkmode bool ;** (noop java) set background mode for gbkcolor to OPAQUE with 0, or TRANSPARENT with 1

**glbmp filename ;** (only win32) display bitmap in glbmpxywh rectangle

**glbmpxywh x y w h ;** (only win32) set rectangle for the display of a bitmap

**glbrush " ;** select solid brush in current color

**glbrushnull " ;** select null brush (leaves area painted with it unchanged)

**glcapture type ;** type is 0 release, 1 capture, 2 line band, 3 box band, 4 ellipse band, 5 vertical line, 6 horizontal line. A capture is normally done in a mouse down event and a glcapture 0 should be done in the mouse up event.

**glcaret w h ;** (not java) create a text caret at the current position with a width and height of w h.

**glchord x y w h xa ya xz yz ;** (only win32) same parameters as garc. Drawn with pen and brush.

**glclear " ;** clear isigraph drawing buffer and reset defaults. If a printer context has been created, it is carried forward. Default values are 0 except for the following:

```
glmap MM_DEFAULT
glzmap twips
glbkcolor white
glwindowext 1000 1000
glplaymap anisotropic
glplayextent 1000 1000
```

glxextent 1

**glclear n** ; initialize graphics state. If n is " or 0, then an InvalidateRect is done, If n is 1, then it is not done.

**glcursor n** ; sets mouse cursor. Values are defined as IDC\_... in gl2.

**glellipse x y w h** ; draw an ellipse in the rectangle with pen and brush

**glxextent bool** ; 0 selects the screen context and 1 selects the printer for glqextent and glqtextmetrics. The default is 1.

**glxextentfont [fontspec p132](#)** ; set font for glqtextent and glqtextmetrics

**glfile filename** ; (only win32) set filename for glsave and glsavebmp. " is the clipboard.

**glflood i j color** ; (only win32) fill area around point i,j with the current brush. The area is bounded by color.

**glfont [fontspec p132](#)** ; font for text commands

**glgrid "** ; buffered drawing command that causes a grid to be painted. The grid is painted based on the state information set by the following glgrid... commands.

**glgridatt attributes** ; A set of attributes is a list of 20 integers. The argument is the ravel of the sets of attributes.

attribute set: pen, brush, text, font, align, edit, 8 unused values

pen, brush, and text are 3 integers with RGB values

font: \_1 selects font based on first character (-, \_ , and digit select font 1); 0 selects font 0, 1 selects font 1

align: \_1 selects left or right based on first character (-, \_ , and digit selects right), 0 selects right, 1 selects left, 2 selects center.

edit: 0 not editable, 1 editable

**glgridborder borders** ; a border is specified as 20 integers:

row, col, rows, cols, left b, top b, right b, bottom b

b field for each side is a type and RGB color  
type is 0 to 3 and 11 for none, thin, medium, fat, and double line

**glgriddrawmark "** ; undraw previous mark and draw new mark

**glgridedit text** ; create edit box at mark

**glgridedit 0** ; destroy edit box

**glgridedit 1** ; enable arrow keys in edit box

**glgridfill width height fixfilltype filltype** ; information about cells beyond the grid data

**glgridfix row col** ; fixed rows and columns

**glgridfont0 [fontspec p132](#)** ;

**glgridfont1 [fontspec p132](#)** ;

**glgridgetedit "** ; get text from edit box

**glgridgettext row col** ; get text for this cell

**glgridgettype row col** ; get type and attributes for this cell

**glgridh heights** ; row heights for the grid data

**glgridinvalidate bool** ; 1 causes subsequent glgridtext and glgridtype commands to invalidate the cells they change.

**glgridmark row col rows cols** ; marked range of cells

**glgridrc rows cols** ; rows and columns in grid data

**glgridrchw row col rows cols** ; select subarray used in subsequent glgridtext and glgridtype commands

**glgridscroll row col** ; scrolled rows and columns

**glgridskip row col** ; rows and columns in total data that are not in grid data

**glgridtext** ; NULL terminated strings of grid data

**glgridtype types** ; integer value for each cell that selects its attributes

**glgridw widths** ; column widths for the grid data

**glline i j** ; draw line from current position to point i,j and update current position

**gllined I j k l linelen spacelen [linelen spacelen ...]** ; (not java) draw dashed line from i,j to k,l. Works only for vertical and horizontal lines.

**gllines pts** ; draw connected lines. pts is 2 or more points.

**glmap mode** ; See [mapping mode p134](#).

**glmark** ; (not java) set mark in graphics buffer and reset pens etc.

**glmarke** ; (not java) clear graphics buffer back to last gmark (does not undraw)

**glmove i j** ; move current position to i,j

**glnoerasebknd bool** ; default paint clears background and if commands paint entire area this is not necessary. Set to 1 to avoid initial paint of background.

**glpaint "** ; does immediate paint of a 2D control. The painting of a 3D control is delayed until the paint event handler can be run.

**glpaintx "** ; mark control for painting when possible. If a glpaintx is done and J continues execution for a while, then the painting will be delayed.

**glpen i [style]** ; select pen. pen is color from last glrgb command and is i units wide. style is from the set (ps\_solid, ps\_dash, ps\_dot, ps\_dashdot, ps\_dashdotdot, ps\_null, ps\_insideframe)

**glpie x y w h xa ya xz yz** ; (not wince) draw pie shaped wedge with pen and brush. Same parameters as garc.

**glpixel i j** ; draw pixel at i,j in current color

**glpixels x y w h pixeldata** ; pixeldata is an integer per pixel with RGB values



**glplay filename ;** (only win32) the wmf file filename is played into the isigraph control. The glplaymap, glplayxywh, and glplayext commands determine how the file is played.

**glplayext extx exty ;** (only win32) windowport extent information for scaling the wmf file

**glplaymap mode ;** (only win32) set map mode for playing a wmf file.  
See [mapping mode p134](#).

**glplayxywh x y w h ;** (only win32) set rectangle for playing a wmf file. This is not a clipping rectangle and if the drawing goes beyond the w and h values it will be displayed. The w h values establish the extents that are used for scaling if the drawing is done in an isotropic or anisotropic mapping.

**glpolygon pts ;** draw polygon in pen and brush

**glpolymode bool ;** (only win32) 0 selects **alternate** filling of polygon and 1 selects **winding**

**glqdevmode " ;** (only win32) return printer document properties (see glzdevmode)

**glqextent text ;** return width and height of the text in a particular font on a particular device. The glqextent command determines whether the screen or printer context is used and glqextentfont determines which font is used.

The layout (rasterization) of text is quite complicated and the final length of text is not simply the sum of the lengths of the individual characters. For this reason, calculations based on character widths are not very useful.

In addition, the length of text depends on the device. Drawing "testing 1 2 3" with an arial 240 font in a twips mapping will have different lengths on the screen, on a 150dpi printer, and on a 300dpi printer. Calculations such as line breaks, page breaks, and positioning stuff at the end of text are sensitive to the actual device.

**glqpixels x y w h;** return pixeldata as integer per pixel with RGB values

**glqprinter "** ; (only win32) return printer information

horzsize	width in millimeters
vertsize	height in millimeters
horzres	width in pixels
Vertres	height in raster lines (pixels)
logpixelsx	horizontal pixels per logical inch
logpixelsy	vertical pixels per logical inch
bitspixel	number of color bits per pixel.
planes	number of color planes
numcolors	number of entries in the device's color table (_1 if > 8)
aspectx	relative width of a pixel
aspecty	relative height of a pixel
aspectxy	diagonal width of a pixel
physicalwidth	width of the physical page. A 600 dpi printer on 8.5"x11" paper could have a physical width of 5100. The physical page is almost always greater than the printable area of the page, and never smaller.
physicalheight	height of the physical page
physicaloffsetx	distance from the left edge of the physical page to the left edge of the printable area. A 600 dpi printer on 8.5"x11" paper, that cannot print on the leftmost 0.25" of paper, has a horizontal physical offset of 150.
physicaloffsety	distance from the top edge of the physical page to the top edge of the printable area

**glqtextmetrics "** ; return font information. The context (screen or printer) is selected by glxextent and the font is selected by glxextentfont. The values are: Height, Ascent, Descent, InternalLeading, ExternalLeading, AverageCharWidth, MaxCharWidth

**glqwh "** ; return window width and height in pixels.

**girect x y w h** ; draw rectangle with pen and brush

**glrgb color** ; set current color

**glroundr x y w h rw rh** ; (not java) draw rectangle with rounded corners defined by ellipse with width rw and height rh

**glsave flip w h** ; (only win32) save drawing in WMF format in the glfile filename. Save to the clipboard if filename is empty.

flip 0 has J orientation of 0 0 as lower left corner.

flip 1 flips for applications like Word with 0 0 as upper left corner.

w h are suggested clipboard width and height in 0.01 millimeters.

The wmf file is saved without a "placeable metafile header". Some applications require this 22 byte header at the front of a wmf file.

The utility `addwmfheader` from script `system\main\winutil.js` adds a header to a wmf file so that it is suitable for use by WORD and similar applications.

**glsavebmp width height** ; (only win32) save drawing as bitmap (24bit color) to glfile filename. Save to clipboard if filename is empty.

**glsel id** ; select isigraph child of currently selected parent for graphic commands

**glsel hwndc** ; hwndc is the result of `wd'qhwndc'` for an isigraph control. Parent and child are selected as the wd and gl command targets.

**glshow "** ; add new commands (since last `glshow` or `glshowx`) to the 2D paint buffer and immediately paints just the new commands. A `glshow` immediately after creating and showing a form can result in extra painting and flicker. The `glshow` paints the new stuff, which in most cases is everything, and then the normal window painting mechanism requests painting the new window and the painting will be done again. It is generally better to use `glshowx` and `glpaintx`.

**glshowx "** ; add new commands to the 2D paint buffer with no explicit painting

**gltext text** ; write text in the glfont font. Where and how the text is displayed is affected by the gltextalign and gltextxy commands.

**gltextalign TA\_value** ; the TA\_value affects how gltext displays text

TA\_LEFT TA\_CENTER TA\_RIGHT  
TA\_TOP TA\_BASELINE TA\_BOTTOM  
TA\_NOUPDATECP TA\_UPDATECP

The first option in each group is the default. TA\_NOUPDATECP causes gltext to use the position set by gltextxy. TA\_UPDATECP causes gltext to use and update the current position.

Values from each of the 3 groups can be combined:

gltextalign TA\_BOTTOM + TA\_UPDATEDCP

**gltextcolor " "** ; glrgb color is set as color of text for gtext

**gltextjustify space nblanks**; (only win32) space is the amount of space to insert for nblanks. This space per blank is inserted for EACH blank in subsequent gtext commands. Because of rounding error there may be unused space at the end of the text. To allow spacing multiple texts on the same line this unused space is carried over into the next qlgextent calculations. Justifying a line should be sure to start with this unused space set to 0. A gltextjustify 0 0 sets this unused space to 0. Each section of a line should be measured with glqextent and then a gltextjustify should be done with the required space and the number of blanks in that section of text, and finally the gltext should be done. At the end of the justifying a line it is a good practice to do a gltextjustify 0 0 to clear the unused space value and to be sure that subsequent text is not inadvertently spaced out.

gltext commands painted with glshow after gltextjustify commands have been processed will not be justified unless the entire control is repainted. glpaint forces a repaint.

The Macintosh runs the gltextjustify command without error, but text justification is not supported and the text is displayed without justification.

**gltextxy x y** ; position for gltext with gltextalign of TA\_NOUPDATECP

**gltop topvalue ;** (not java) topvalue is in logical units and ties that y coordinate value to the top of the drawing . This can be used to make a window display the top part of an image, rather than the bottom part.

**glwantresize ;** request resize event for gl2 control.

**glwindowext x y ;** default is 1000 1000. Change these values to change the scale (zoom) in anisotropic and isotropic mappings

**glwindoworg x y ;** (not java) default is 0 0. Change these values to scroll the drawing in the viewport

**glzabortdoc " ;** (only win32) cancel the print job

**glzcreate printrname ;** (only win32) name from the set of printers returned by wd qprinters. If no name is given then the session default printer is used. If a printer context already exists, it is deleted before creating the new context. The session default printer is the system default printer when J is started. This can be changed with wd mbprinter.

**glzdelete " ;** (only win32) deletes printer context created by glzcreate. There is no error if there is no context.

**glzdevmode " ;** (only win32) modifies printer document properties such as orientation and copies. The command glqdevmode returns the properties that can be changed by glzdevmode and their current values. You can determine arguments for glzdevmode by using mbprinter to create the desired printer document setup and then use glqdevmode to get those values. This command must be done before the glzstartpage command.

The Macintosh does not support glzdevmode and printer properties can not be changed.

The information returned is: orientation, papersize, paperlength, paperwidth, scale, copies, source, quality, color, duplex, yresolution, truetype

**glzenddoc " ;** (only win32) ends a document.

**glzendpage "** ; (only win32) ends a page.

**glzmap mode** ; (only win32) set mapping mode used in printing. Default is twips.

**glzprint "** ; (only win32) draws the image on the printer.

**glzprintpage** ; (only java) prints current page

**glzstartdoc 'jobname [filename]'** ; (only win32) the job name can be up to 31 characters and appears in print manager. The filename indicates to print to a file instead of the printer. This file can be printed with the print command. The Macintosh does not support the second parameter of file name.

```
glzstartdoc "first job"  
glzstartdoc "my job" tofile.prn'
```

**glzstartpage "** ; (only win32) starts a page. Each page must be bracketed by a glzstartpage and a glzendpage.

**glzstartprint** ; (only java) start printer job

**glzorientation n** ; (not wince) 0 landscape, 1 portrait

## fontspec

A fontspec is several parameters that specify a font. A fontspec is used in some wd, gl2 and gl3 commands.

A fontspec is a font name, size, and and optional styles:  
name size [italic] [bold] [underline] [oem] [angle]

For example: "lucida console" 15 italic

A positive size gives cell height and negative gives character height. In both cases a font defined interline spacing is used. It is recommended to use cell height (positive) as this maps more directly and accurately to physical fonts. This is particularly true for printers.

The size is in logical units. For wd commands this is generally point size. In gl2, size is in mapping mode units. In gl3, size is in pixels.

Fonts can have different character sets. The standard Windows character set is ANSI and does not include boxes. The old PC character set is OEM and does contain boxes. Fonts such as Courier New and MS Linedraw contain a single DEFAULT character set. The oem parameter selects an OEM font. The default parameter selects the DEFAULT font. If neither oem nor default is supplied, an ANSI font is selected. If there is not an exact match, you get an arbitrary font. There are differences in distributed fonts and in the way fonts are selected in different hosts.

isij 12 default	isij with boxes
"ms linedraw" 12 default	linedraw with boxes
"lucida console" 12 oem	console with boxes (if the font has oem)
"courier new" 12 oem	courier with boxes (if the font has oem)

You may have to experiment to find the best font for J boxes with your particular host, fonts, screen, and printer.

Fonts such as WINGDINGS, do not have an ANSI character set, and you need to specify default to get the default character set for the font.  
`glfont 'wingdings 100 default'`

The angle style gives an angle in 10ths of degrees clockwise from the baseline. For example:  
`glfont 'arial 240 angle900'`



## isigraph events

An isigraph control supports character and mouse events. The Code Dialog in the form editor lists all events for an isigraph control.

char                                      isij with boxes

sizeresize event (OpenGL, gl2 must request with glwantresize) paint

paint event (OpenGL only) mmovemouse move event

mbldblmouse button left double-click mblmousedown button left down  
mblupmouse button left up

mbrdblmouse button right double-click mbrmousedown button right down  
mbrupmouse button right up

For mouse events the wdhandler variable sysdata contains:  
x y width height leftbutton rightbutton ctrl shift

The x y width and height are in pixels and the other values are 1 if the corresponding button or key was down when the event occurred.

For a char event sysdata contains the value of the character. Characters, such as HOME, END, or the arrow keys are returned as 128+VK\_name (virtual key) as defined in packages\graphics\vkeys.js.

## Mapping Mode

gl2 command values are in logical units and the mapping mode affects how logical units are mapped onto the display surface. The mapping modes are:

```
MM_DEFAULT      lower left 0 0, upper right 1000 1000
MM_RIGHTDOWN    upper left 0 0, lower right 1000 1000
MM_RAW          upper left 0 0, values are pixels
MM_TWIPS        upper left 0 0, values are twips (win32 only)
```

glmap sets the mapping mode.

### MM\_DEFAULT

The default is a logical drawing surface that has 0 0 as its lower left corner and 1000 1000 as its upper right corner. The x axis starts at 0 and is positive to the right. The y axis starts at 0 and is positive upwards.

The **windowport** is the drawing represented by the drawing commands. The **viewport** is the surface (screen window or printer paper) that the drawing is drawn on.

The scaling is based on the ratio between the x and y extents of the viewport and windowport. It is the ratios of these extents that is important, not the individual values.

The viewport (drawing surface) origin is fixed as 0 0 in the lower left corner. The viewport extent is fixed as the pixels in the x and y directions. These values can not be changed by commands. However, as the window on the screen is resized, the viewport extents change.

The default windowport (drawing) is 0 0 and the default windowport extent is 1000 1000. The glwindoworg command changes the origin and the glwindowext command changes the extent.

### MM\_RIGHTDOWN

This is the same as MM\_DEFAULT except the y axis is flipped. The upper left

corner is 0 0 and the lower right corner is 1000 1000. The x axis starts at 0 and is positive to the right. The y axis starts at 0 and is positive downwards.

## **MM\_RAW**

Values are in pixels and the drawing is directly to the pixels on the drawing surface. The upper left corner is 0 0 and x goes to the right and y goes down. The resize event (glwantresize) can be used to redraw based on the new pixel size of the control.

## **MM\_TWIPS**

This mode is supported only in JFEwdw (win32 only). It is similar to MM\_RAW except that the logical units are in twips and are scaled to the device. A twip is a "twentieth of a point". A point is approximately 1/72 of an inch and in computer systems a twip is considered to be exactly 1/1440 of an inch. This mode is particularly suited to laying out text and graphs for printing.

## gl3 commands 11!:3000+n

gl3 commands are for 3D (OpenGL) drawing in an isigraph control. Script system\main\gl3.ijs defines gl3 verbs.

load 'gl3' NB. defines verbs such as: glAccum=:11!:3001

The syntax of most OpenGL verbs should be clear from standard OpenGL API documentation, even though that documentation is oriented towards C programmers.

You are encouraged to start learning OpenGL programming with the **OpenGL Introduction** lab from the **Studio|Labs** menu command.

This section documents the gla... commands that are not part of the OpenGL specification or where the syntax needs clarification. The gla... commands are concerned with how OpenGL fits into J or with Windows extensions.

Most 2D commands do not work in a 3D isigraph control. The following are the commands that do work: glshow, glshowx, glpaint, glpaintx, glqwh, glcapture.

[OpenGL Printing p136](#) shows how to print 3D images.

## gl3 commands

glFont [fontspec p132](#) - font for glaUseFontBitmaps and glaUseFontOutlines

glGetErrors " - return recent errors reported by Quadric, Nurbs, and Tess error callbacks. The error callbacks must be enabled with the appropriate gluQuadricCallback, gluNurbsCallback, or gluTessCallback.

glARC " - create OpenGL render context for drawing on an isigraph window. This must be run before pcenter, pmove, or any other positioning is done.

Fails if:

control is in use by 2D graphics

unable to initialize the OpenGL dlls (opengl32.dll and glu32.dll)

control missing `ws_clipchildren` and `ws_clipsiblings` styles

`glaRC` type `w h` - create OpenGL render context for drawing on a bitmap: type must be 1. `w h` give width and height of a 24bit color bitmap.

A bitmap RC is associated with an isigraph window but doesn't display anything in the window so normally this window should be hidden.

If the window is hidden, the `form_isigraph_size` and `form_isigraph_paint` handlers are not automatically called and they must be called explicitly.

The size handler for a bitmap RC should not use the window size (`glqwh''`) for setting the viewport. It should use the width and height used to create the bitmap in `glaRC`.

`glaSaveBMP filename` - save bitmap of an isigraph control with a bitmap RC. If the filename is "", the bitmap is saved to the clipboard.

`glaSwapBuffers` - the front-buffer is the one displayed on the screen. Drawing is done into the back-buffer. A `glaSwapBuffers` makes the back-buffer the front-buffer.

`glaUseFontBitmaps 0 first count listbase` - create display lists based on `glaFont`. A display list is created for count characters starting at first. The display list numbers start at listbase.

`glaUseFontOutlines 0 first count listbase deviation extrusion format` - create display lists for drawing 3D characters based on `glaFont`.

first - first of the set of characters

count - number of characters used to create display lists

listbase - starting display list

deviation - maximum chordal deviation from the original outlines. When deviation is zero, the chordal deviation is equivalent to one design unit of the original font.

The value must be equal to or greater than 0.

extrusion - how much a font is extruded in the negative *z* direction. The value must be equal to or greater than 0.

format - `WGL_FONT_LINES` or `WGL_FONT_POLYGONS`

Returns a matrix with a row for each character. The row contains:

BlackBoxX - black box (smallest rectangle that contains the glyph) width

BlackBoxY - black box height

OriginX - black box upper-left x coordinate

OriginY - black box upper-left y coordinate

IncX - horizontal distance from current cell origin to the next cell

IncY - vertical distance from current cell origin to the next cell

glBitmap - argument is boxed list where the last argument is the data argument

glDrawPixels - argument is boxed list where the last argument is the data argument

glCallLists integer\_data

glCallLists character\_data

glClearColor R G B [A] - sets clear color red, green, blue, alpha. If A is elided, it is set to 1.

glColor R G B [ A ] - sets color red, green, blue, alpha. If A is elided it is set to 1.

glFeedbackBuffer integer type - feedback buffer size in floating values. This command is only allowed once. This avoids potential crashes with the buffer being changed at the wrong time.

glGetError " - return OpenGL error flags (there may be more than one).

gluErrorString returns a string for an error number.

glPixelStore - GL\_UNPACK\_ALIGNMENT and GL\_PACK\_ALIGNMENT are the only pname arguments supported.

glSelectBuffer size - select buffer size in integers. This command is only allowed once. This avoids potential crashes with the buffer being changed at the wrong time.

The result data from the select buffer is recorded for commands done after a glRenderMode GL\_SELECT. The result buffer is returned as the result of a glRenderMode any\_other\_mode. OpenGL defines the select buffer as returning minimum and maximum z values as unsigned integers.

unsigned int value - J signed int value

0 - smallest negative int

max 32 bit unsigned int - largest positive int

To convert the J signed integer range to a float range from 0 and up, add  $2^{32}$  to the negative integers.

glTexImage1d - argument is boxed list where the last argument is the data argument

glTexImage2d - argument is boxed list where the last argument is the data argument

glVertex X Y [ Z [ W ] ] - sets vertex coordinates. If W (rarely used scaling value) is elided it is set to 1. If Z is elided it set to 0.

gluBuild1dMipmaps - argument is boxed list where the last argument is the data argument

gluBuild2dMipmaps - argument is boxed list where the last argument is the data argument

gluPwlCurve nurb type data - count and stride parameters are fixed based on type and length of data.

gluNurbsCurve nurb uknots udata ustride uorder type data

gluNurbsCurve nurb uk ud vk vd ustride vstride uorder type data

gluScaleImage - argument is boxed list where the last argument is the data argument

## OpenGL printing

Printing OpenGL images is not directly supported. However, it is possible to get reasonably good printing of OpenGL images by creating a bitmap RC, drawing to the bitmap, and then printing the bitmap. If the bitmap is not stretched to much to fit the printing area the quality of the printed image is good.

The steps are as follows:

1. create temporary form and isigraph control
2. create a bitmap RC
3. size image - the size handler, but with bitmap size, not glqwh"
4. draw image - the paint handler
5. save the bitmap to file

Something like the following could be used to print an OpenGL image drawn on form a in isigraph control.

```
print=: verb define
wd 'pc p;cc g isigraph ws_clipchildren ws_clipsiblings;'
BMPSIZE=.2$1000 NB. big bitmap for little stretch
glaRC 1,BMPSIZE
NB. do initialization required by size and paint
size BMPSIZE NB. a_g_size calls size with glqwh''
a_g_paint''
glaSaveBMP 't.bmp'
wd'pclose'
NB. print t.bmp file with 2D printing
)
```



## **Form Editor**

[Overview p138](#)

[Hints p139](#)

[Mouse p140](#)

[Keyboard p141](#)

[Design p142](#)

[New Control p143](#)

[Control p144](#)

[Parent p145](#)

[Menu p146](#)

[Toolbar p147](#)

[Statusbar p148](#)

[Code p149](#)

[Tab Order p150](#)

[Run p151](#)

[Defaults p152](#)

[Tech Notes p153](#)

## Form Edit Overview

A form definition is the set of wd commands that define a parent window and its controls. The Form Editor takes a form definition from a script file, displays the form, supports editing, and saves the new definition back in the script. The script also contains the verbs that handle events when the form is run. The Form Editor facilitates editing these verbs.

The best way to learn about this is to try it. Create a new script and run Edit/Form Edit. Click the Design New button to add controls. Drag controls with the mouse.

Controls can be marked with red or blue borders. The red control is the selection and all marked controls are the collection. The mouse marks and unmarks controls.

[Hints p139](#)

[Mouse p140](#)

[Keyboard p141](#)

[Design p142](#)

[New Control p143](#)

[Control p144](#)

[Parent p145](#)

[Menu p146](#)

[Toolbar p147](#)

[Statusbar p148](#)

[Code p149](#)

[Tab Order p150](#)

[Run p151](#)

[Defaults p152](#)

Tech Notes

## Form Edit Hints

Pick a good form id at the outset, to avoid having to change it later.

Create a new script and save it as a permanent file. The form id might be a good name for the script.

Menu command Edit/Form Edit activates the form editor. It asks for a form id and a template.

Use the [Design Dialog p142](#) to create, position, and size controls. The New button activates the [New Control Dialog p143](#) used to add new controls to the form.

In dialogs such as New Control, OK performs the action and returns to the Design Dialog, and Apply performs the action and stays in the dialog.

Ctrl+C copies the collection to an internal clipboard and Ctrl+V, Ctrl+B, or Ctrl+N paste. For example, create a radiobutton, move it, Ctrl+C, and do Ctrl+N several times.

The Tab key can be used to find a control covered by another control.

You can open more than one form and copy and paste between forms.

Create the controls you want and position and size them roughly. Fine tune position, size, and alignment, after everything is on the form.

Customize control styles, captions, and ids with the [Control Dialog p144](#). Give descriptive ids to controls that cause events or will be used in wd commands.

Create a menu with the [Menu Dialog p146](#).

Set the [tab order p150](#) after the form is pretty much complete. Moving, adding, or deleting controls require redoing the tab order. Radiobuttons are automatically grouped if they are in the right tab order.

[Run p151](#) the form with the Design Dialog Run button. This shows the form as it

will look to the user. Check the tab order and test your event handlers.

The [Code Dialog p149](#) shows which verbs are required to handle events and allows you to activate the form script and position the caret at the definition. Do not add handlers until the layout is settled.

Ctrl+click a control, a menu item, or the script to switch between the script and the form editor.

## Form Edit Mouse Actions

click -- select control (collection is maintained)

shift+click -- toggle mark on control

ctrl+click -- activate script (ctrl+click script to return)

double-click -- show Control Dialog

drag move -- drag collection. Ctrl+shift to move just the selection.

drag resize -- just outside the red border shows a resizing cursor. Ctrl+shift to resize just the selection.

draw select -- press the mouse outside any control and draw a box to select controls

A mouse action in a marked control affects that control. To select a control covered by a marked control, first unmark the control. The mouse selects the control with the nearest upper left corner.

The mouse displays the form menu. Select a menu item with Ctrl to activate the script.

If [tab order p150](#) is displayed mouse actions are different.

## **Form Edit Keyboard Actions**

Ctrl+X -- cut collection (internal clipboard)

Ctrl+C -- copy collection

Ctrl+V -- paste

Ctrl+B -- paste Beside collection

Ctrl+N -- paste beNeath collection

Ctrl+T -- toggle [tab order p150](#) display

Del -- delete collection

Tab -- select next in tab order

Shift+Tab -- select previous

## Form Edit Design Dialog

The New button shows the [New Control Dialog p143](#) for adding controls to the form.

Move collection with the cross direction buttons. The central button switches the move-grid between 1 and 4. The move is to the move-grid if not already on the grid.

Align collection with shift + a direction button.

Size collection with ctrl + a direction button.

The shift key causes the following buttons to act down rather than across.

Center -- center collection

Size -- size collection to the selection width

Space -- space 3 or more controls across the form

Touch -- space collection so they touch

Minus -- remove space

Plus -- add space



## **Form Edit New Control Dialog**

Design Dialog New button shows this dialog that is used to add controls to the form.

Select the type of control you want to create and give the control an id.

Some controls, such as edits, are often labeled by a static. If you want to label the control, fill in the label field. Check down if you want the control below the label.

To create several controls of the same type, for example several radiobuttons, fill in the copies field. Check down if you want them vertical.

Click OK or Apply to recreate the form with the new control(s).

## **Form Edit Control Dialog**

Double-click a control to show this dialog. Make changes and press OK or Apply to recreate form.

The listboxes can have multiple items selected.

Ownerdraw buttons are not drawn when designing the form. The file name is in the caption. Edit the caption to be the name of the file with the graphic for the button.

## Form Edit Parent Dialog

Design Window/Parent shows this dialog. Make changes and press OK or Apply to recreate form.

Styles:

closeok -- parent closes without causing an event

dialog -- heavy frame

nomax -- no maximize button

nomenu -- no system menu

nomin -- no minimize button

nosize -- no sizing frame

owner -- owned by currently selected form (owner disabled until close)

Options do not affect the form during design.

The pmove button toggles use of a pmove command. When set, the current design location is shown to the right of the button. The form will be moved to that location when it is created.

The right and bottom margins set the margins used in creating the form.

# Form Edit Menu Dialog

Design Window/Menu shows this dialog. Edit the definition and press OK or Apply to recreate the form.

A popup menu item definition has just a caption. For example:  
File

A command menu item definition has an id, caption, and optional shortcut, tooltip, and statushelp. For example:  
idopen caption \_ shortcut \_ tooltip \_ statushelp

The \_ character separates fields that may contain blanks. The shortcut appears at the right side of the menu list. The tooltip does not affect the menu, but a toolbar button with the same id will show the tooltip when the mouse rests on the button. The statushelp shows in the statusbar when the mouse is on the menu item.

+ ends a popup menu, - is a separator. and & underlines the next character.

For example:

&File

new &New

open &Open \_ Ctrl+O \_ tool tip \_ status help text

-

idquit Quit

+

&Help

idhelp &Contents

Use the mouse to show the menu on the form. Press Ctrl and select a menu item to go to the code in the script for that event.

Use the [Code Dialog p149](#) to define verbs for menu ids and shortcuts.

A menu shortcut usually calls the verb defined for the menu id. For example, shortcut Ctrl+O could define a handler for fkey 'o ctrl' that called `formid_idopen_button`.

## Form Edit Toolbar Dialog

Design Window/Toolbar shows the Toolbar Dialog. This dialog manages the form toolbar.

A toolbar is a set of buttons that show across the top of form, below the menu.

The buttons are drawn from a bitmap. The File button allows you to select the bitmap for your toolbar. The selected bitmap shows across the top of the dialog with the buttons numbered.

A line in the edit box gives the id for the toolbar button, the index of the bitmap picture to use, a tooltip to display when the mouse rests on the button, and statushelp to display in the status bar.

A line with just an index is a separator where the index is the width.

For example:

idnew 0 new file \_ Create a new file.

idsave 2 save file \_ Save file.

25

idprint 6 print file \_ Print file

A toolbar button is usually a shortcut for a menu item and in this case they should have the same id.

## Form Edit Statusbar Dialog

Design Window/Statusbar shows the Statusbar Dialog. This dialog manages the form statusbar.

A statusbar is a set of panes across the bottom of a form.

The first pane is the help pane and its width varies from a minimum up to the space remaining. It has default text set by the sbarset command. It shows menu and toolbar statushelp when the mouse is on a menu item or toolbar. When this temporary help is not shown the pane reverts to its default contents.

A line in the edit box give the id, width, and initial text for each pane in order.

For example:

idstatushelp 100 Press F1 for help.

idstatusrdy 50 Ready

idstatusoption 75 option

## Form Edit Code Dialog

Design Window/Code shows this dialog.

This dialog helps you define and edit verbs in the script that handle events.

The button event type lists the ids of controls that cause button events.

Items are indented if a verb is not defined for that event.

Double-click an item in the listbox to activate the script with the caret located at the verb. If there is no definition, a default definition is added. Ctrl+click the script to return to the form editor.

The fkey event type lists keys that cause events. Alphabetic and numeric keys with Ctrl or Ctrl+Shift as well as function keys cause events. Fkey events Ctrl+Z,X,C, or V are not supported. Menu shortcuts are handled as fkey events.

The other event type lists general form events.

The orphan event type lists verbs defined in the script for which there is no control that would cause the event. For example, create a button and create a handler verb in the script, then delete the button.

Ctrl+click a control, a menu item, or the script to switch between the script and the form editor.



## **Form Edit Tab Order**

Ctrl+T in the form toggles display of the tab order.

The tab order determines how the user moves around on the form with the tab key.

Mouse actions with the tab order displayed are as follows:

click -- select control (red border)

shift+click -- put new selection after previous selection in the tab order

ctrl+click -- put new selection as first in tab order

## Form Edit Run

Design Dialog Run button saves the form in the script, saves the script, runs the script, and runs the verb formid\_run to create and show the form.

[Code Dialog p149](#) other event type entry run takes you to the form run verb definition.

## **Form Edit Defaults**

Design Dialog File/Defaults displays the sizes used for controls created with New Control. The defaults can be changed and can be recorded in the ini file.

The default form font can be set by the main J menu with View/Set Form Font. Use File/Save jsm.ini to save a new form font as the default in the ini file.

## **Form Edit Tech Notes**

A form definition in a script begins with a line that starts with a pc command and ends with the line <rem form end>.

Scripts in system\packages\forms are used as templates for new forms.

## **Regular Expressions**

[Regular Expression p155](#)

[Patterns p156](#)

[Verbs p157](#)

[Utilities p158](#)

[Demo p159](#)

[Copyright p160](#)

## Regular Expression

A regular expression is a text string that specifies a pattern of characters. This facility allows you to write J programs using regular expressions to search arbitrary text. You can perform a search for a single or multiple matches, or extract the matched text from the string. You can also merge new text in to replace the matches, or apply a verb to the matched text in a string.

Utility verbs are included to assist in building regular expression patterns, as is a utility to search J scripts and other text files for patterns.

The primary definitions are in `system\main\regex.ijs` ; utility verbs to build patterns are in `system\packages\regex\regbuild.ijs` .

The Find in Files utility, available from the Edit menu or by pressing Ctrl+Shift+F, can search for simple text, regular expression, or special patterns such as the assignment of a name.

Labs and a demo on regular expressions are available from the **Studio** menu.

## Regular Expression Patterns

A regular expression pattern is a sequence of elements which matches successive portions of a character string. For example, simple letters are elements which match the same characters in the string. The asterisk indicates that the previous element should be matched 0 or more times. So, a pattern of `abcd` must match in the string exactly; a pattern of `ab*cd` matches the letter `a` followed by 0 or more occurrences of the letter `b`, followed by the letters `cd`. The particular elements of a pattern are described below.

### Characters

Non-special characters match exactly. Non-special characters are anything other than: `[ ] ( ) { } $ ^ . * + ? | \`

A special character is included as simple text by preceding it with a backslash.

### Character sets

The special character `.` matches any character (except the null character, `0{a. }`)

The special characters `^` and `$` match the start and end of lines.

Sets of characters are defined by enclosing the list of characters in brackets:

`[aeiou]` matches a single vowel character

Ranges can also be included within the brackets:

`[a-z]` matches any lower case letter

Combinations of the above are acceptable:

`[a-zA-Z13579]` matches any lower case, upper case, or odd digit

Fixed sets (classes) of characters can be included in the list, as a name within bracket-colon pairs:

`[#:digit:]abc]` matches the character `#`, a digit, or any of the letters `a`, `b`, or `c`

The character classes defined are:

alnum

alphanumeric

alpha

alphabetic

blank

tab and space

cntrl

control chars

digit

digits

graph

printable (except space)

lower

lowercase

print

printable

punct

punctuation

space



whitespace

upper

uppercase

xdigit

hex digits

If a set begins with  $\wedge$ , then the pattern will match with any character *not* in the set.

### Subexpressions

A series of elements may be combined by enclosing them in parenthesis.

Subexpression are affected by closures such as  $*$  just as simple characters are:

$([a-z][0-9])^*$  matches any number of occurrences of a letter followed by a digit

The result of searches for a pattern return a match for the overall pattern, and a separate match for each subexpression

A  $\backslash$  followed by a digit, N, matches the same substring which occurred in the Nth subexpression:

$([[:digit:]]^+)\#\backslash 1$  matches one or more digits, followed by a #, followed by the same string of digits

### Closures

A  $*$  following an element matches 0 or more occurrences of that element:

$[aeiou]^*$  matches 0 or more vowels

A  $+$  following an element matches 1 or more occurrences of that element:

$[[:alpha:]]^+$  matches 1 or more alphabetic characters

A  $?$  following an element matches 0 or 1 occurrences of that element:

$-?[[:digit:]]^+$  matches an optional hyphen, followed by 1 or more digits

An interval expression,  $\{m,n\}$ , follows an element to allow it to match at least m, and no more than n, occurrences of the element:

$[[:digit:]]\{3,5\}$  matches 3, 4, or 5 digits

### Alternation

Multiple regular expressions can be separated with a vertical bar `|` to match any of them:

`print|list|exit` matches any of the strings `print`, `list`, and `exit`

## Matches

When searching for a pattern in a string, it is possible to find multiple substrings which match the pattern. The one that is returned is the one which starts earliest in the string. If more than one match starts at the same place, the longest one is returned.

Even once a particular match is located, it is possible for there to be multiple combinations of the contents of the subexpressions which make it up. As a rule, whenever possible the subexpressions which begin earlier in the string will be as long as possible.

The result of a match is a table which describes the match. The first row covers the whole match, and subsequent rows describe where the subexpressions in the pattern match in the string. Each row has two elements: index of the first character of the start of the match, and the length of the match. Any row which doesn't participate in the match is filled with `_1 0`.

## Regular Expression Verbs

The standard regex verbs are defined in `system/main/regex.ijs`. The main verbs are `rxmatch` and `rxmatches`. The former locates the first occurrence of a match in the string; the latter locates all occurrences. Four other verbs create, list, display, and free up compiled patterns: `rxcomp`, `rxhandles`, `rxinfo`, and `rxfree`.

Most of the rest of the definitions either use the `rxmatch` or `rxmatches` verbs, or take the result of them as arguments.

```
match=. pattern rxmatch string
```

Find first match The result of `rxmatch` is a table, each row being an index/length pair. The first row describes the entire match, one row per subexpression follow which describes where each subexpression was found in the string. Where a match does not occur, `_1 0` is returned.

```
matches =. pattern rxmatches string
```

Find all matches `rxmatches` returns a list of tables, with one item per match in the string. The shape of the result is `#matches` by `#subexpr` by 2.

```
phandle =. rxcomp pattern
```

Compile pattern

```
rxfree phandle
```

Release compiled pattern

```
phandles =. rxhandles "
```

Return all pattern handles

```
'nsub pat' =. rxinfo phandle
```

Return #subexprs;pattern

The verbs `rxcomp`, `rxhandles`, `rxinfo`, and `rxfree` allow you to create pattern handles which are simple integers which represent compiled patterns. A handle can be used anywhere a pattern can be and, if used repeatedly, will avoid having to recompile the pattern on each call.

`rxcomp` compiles a pattern and returns a handle.

`rxhandles` returns a list of all existing handles.

`rxinfo` returns information about a handle. It currently returns a boxed list of 1 + the number of subexpressions and the original pattern. The length of the result may be extended (on the right) in the future.

`rxfree` releases all resources associated with a compiled pattern.

```
errtext =. rxerror ''
```

Error text The result of `rxerror` is a text string describing the last error from a regular expression verb.

```
ismatch =. pattern rxeq string
```

1 if entire string matches Returns a 1 if the pattern fully describes the string. (Similar to `= verb`).

```
index =. pattern rxindex string
```

index of match or #string The result of `rxindex` is the index of the first match, or #string if none. (Similar to `i. verb`).

`mask = . pattern rxE string`

`mask: 1's start matchesrxE` returns a boolean mask of length `#string`, with 1's to mark the start of a match. (Similar to `E. verb`).

`sub = . pattern rxfirst string`

`first substring matchrxfirst` returns the substring in the right argument which matches the pattern.

`subs = . pattern rxall string`

`all substring matches` The result of `rxall` is a boxed list of all substrings in the right argument which match the pattern.

`subs = . matches rxfrom string`

`select substrings matchedrxfrom` returns a box containing the substrings described by each index/length pair on the left.

`subs = . matches rxcut string`

`cut into alternating non-match/matchrxcut` returns a boxed list which will match the original string if razed. The items alternate between non-matches and matches, always starting with a non-match.

`newstr = . string rxrplc (pat;rplcstr)`

replace pat with `rplcstrrxrplc` replaces substrings in the left argument. The right argument is a boxed list of the pattern and the replacement text.

```
newstr = . rplcstrs matches rxmerge string
```

merge `rplcstrs` into string `rxmerge` takes a table of matches as an argument, and returns a verb which merges the boxed strings in the left argument into those positions on the right. (Similar to `}` adverb).

```
newstr = . pattern f rxapply string
```

apply `f` to each match `rxapply` applies its verb argument to each of the substring in the right argument which match the pattern in the left argument.

All verbs which take a pattern as an argument can be called with either a character list containing a pattern or pattern handle (an integer resulting from `rxcomp`). For example,

```
'[[:alpha:]]+' rxmatches str    NB. match all sets of letters in str
```

```
handle=. rxcomp                NB. compile pattern into handle  
'[[:alpha:]]+'
```

```
handle rxmatches str           NB. do the match
```

```
rxfree handle                  NB. (once handle is no longer required)
```

## Notes

1. the `rmatch` and `rxmatches` verbs return either a single or list of matches, respectively, with each match being a table of index/length pairs for the match and each subexpression. Other verbs which use the result of `rmatch` or `rxmatches` tend to only use the first row for each match, which represents the entire match.

2. if you're interested in one or more of the subexpressions, it is possible to identify the specific rows of the match which are to be returned by `rmatch` and

rxmatches. If a boxed array is passed rather than a character or numeric pattern, it is a 2-element list consisting of a pattern and a list of the indices of the important rows in a match.

For example, the pattern `'(x+)([[:digit:]]+)'` matches one or more letters 'x', followed by a string of digits, with both the 'x's and the digits being a subexpressions of the pattern. Each match will be returned as a three-row table, describing the entire match, just the 'x's, and just the digits.

```
pat=. rxcomp '(x+)([[:digit:]]+)'  
str=. 'just one xxx1234 match here'  
pat rxmatches str  
9 7  
9 3  
12 4  
  (pat;1 2) rxmatches str    NB. just the 'x's and digits  
9 3  
12 4  
  
pat |. rxapply str          NB. reverse the whole match  
just one 4321xxx match here  
  (pat;2) |. rxapply str    NB. reverse just the digits  
just one xxx4321 match here
```

## Examples

```
pat=. '[[[:alpha:]]][[:alnum:]]_*)' NB. pattern for J name  
str=. '3,foo3=.23,j42=.123,123'   NB. a sample string  
pat rxmatch str                   NB. find at index 2, length  
4  
2 4  
  
pat=. '([[:alpha:]]([[:alnum:]]_*) *=[.;])' NB. subexp is name  
in assign  
pat rxmatch str                   NB. pattern at 2/6; name at 2/4  
2 6  
2 4  
pat rxmatches str                 NB. find all matches  
2 6  
2 4  
  
11 5  
11 3
```

```

    ]phandle=. rxcomp pat      NB. compile
1
    rxcomp '[wrong'            NB. a bad pattern
|domain error: rxcomp
|      rxcomp'[wrong'
    rxerror ''
Unmatched [ or [^
    rxhandles ''              NB. just handle 1 defined
1

    rxinfo phandle            NB.  return (1+#subexp);pattern
+-+-----+
|2|([[:alpha:]]([[:alnum:]]_)*)*=[.;]|
+-+-----+

    phandle rxmatches str     NB. use phandle like pattern
2 6
2 4

11 5
11 3

    phandle rxfirst str       NB. first matching substring
foo3=.

    phandle rxall str         NB. all matching substrings
+-----+-----+
|foo3=.|j42=.|
+-----+-----+

    phandle rxindex&> ' foo=.10';'nothing at all'  NB. index of
match
2 14

    phandle rxE str           NB. mask over matches
0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0

    '['[:digit:]]*' rxeq '2342342'  NB. test for exact match
1
    '['[:digit:]]*' rxeq '2342 342'
0
                                NB. rxfrom selects substring using index/length
pairs

    phandle rxmatch str

```



```

2 6
2 4          NB. entire and subexpression match

```

```

    m=. phandle rxmatches str
    phandle rxmerge str
+-----+-----+
|foo3=.|foo3|
+-----+-----+

```

```

    phandle rxmatches str          NB. all matches
2 6
2 4

```

```

11 5
11 3
    ]m=.(phandle;,0) rxmatches str  NB. entire matches only
2 6

```

```

11 5
    m rxcut str          NB. return alternating non-match/match
boxes
+---+-----+---+-----+-----+
|3,|foo3=.|23,|j42=.|123,123|
+---+-----+---+-----+-----+

```

```

    phandle |. rxapply str          NB. reverse each match
3, .=3oof23, .=24j123,123
    (phandle;,1) |. rxapply str  NB. reverse just name part of
match
3,3oof=.23,24j=.123,123

```

## Regular Expression Utilities

The script `system\packages\regex\regbuild.ijs` contains definitions to for building regular expression patterns.

Many of the verbs below may enclose its argument in parentheses (to make it a subexpression). For example,

`anyof 'abc'` returns `'(abc)*'`.

The argument is only put in parentheses if necessary.

`anyof set 'abc'` is `'[abc]*'`.

The following verbs correspond directly to a feature of the regular expression notation:

<code>set chars</code>	returns set construction for chars
<code>set 'abc'</code>	
<code>[abc]</code>	
<code>not chars</code>	set of non-matching chars
<code>set not 'abc'</code>	
<code>[^abc]</code>	
<code>sub pat</code>	make a subexpression
<code>set 'abc'</code>	
<code>(abc)</code>	
<code>someof pat</code>	pattern matching 1 or more pat
<code>someof 'abc'</code>	
<code>(abc)+</code>	
<code>(min,max) of pat</code>	pattern matchin min up to max of pat
<code>2 4 of 'abc'</code>	
<code>(abc){2,4}</code>	
<code>pat1 or pat2</code>	pattern matching either pat1 or pat2
<code>'abc' or 'd'</code>	
<code>abc x</code>	

pat1 or pat2	pattern matching pat1 immediately followed by pat2
<pre>'action=' by 'move' or 'copy' action=(move copy)</pre>	
sub pat	makes pat a subexpression
<pre>sub 'abc' (abc)</pre>	
bkref refnum	back-reference to a previous subexpression
<pre>bkref 1 \1</pre>	

Some nouns can be used as parts of regular expressions:

white	pattern matching one or more whitespace characters
owhite	pattern matching optional whitespace
sol	pattern matching the start of a line
eol	pattern matching the end of a line
any	pattern matching any character

### **Finally, some miscellaneous verbs**

plain text	returns a regular expression matching the plain text
<pre>plain 'dir j.*' dir j\.\.*</pre>	
pat1 between y	result is elements of y catenated together with pat1 between each
<pre>' *' between 'abc' a *b *c ' *' between 'p1';'p2';'p3 p4' p1 *p2 *(p3 p4)</pre>	
comment nb	add comment to pattern
pattern	

Interpretation of a pattern always stops at the first null character (0{a.}). The nb verb makes use of this by concatenating a null character and comment at the end of a pattern.

```
p=. rxcomp 'some digits' nb '[:digit:]]+'
rxinfo p
+-+-----+
|1|[:digit:]]+ NB. some digits|
+-+-----+
```

```
setchars setpat    returns list of characters matching a set pattern
    setchars '[a-
d[:digit:]]'
0123456789abcd
```

## Character classes

The following nouns are strings which are used **within** sets to specify a character class: alnum, alpha, blank, cntrl, digit, graph, lower, print, punct, space, upper, xdigit

For example, alpha=. '[:alpha:]'

Corresponding nouns, named with a leading uppercase, are patterns specifying a **set** of the character class, for example, Alpha=. '[:alpha:]' NB. (same as set alpha)

## J patterns

The following nouns, defined in packages\regex\regj.ijs, are patterns which match elements of J code:

Jname

matches a J name

Jnumitem, Jnum

matches a J numeric item or array (constant)

Jchar

matches a J character string

Jconst

matches a J numeric or character constant, include a. and a:

Jgassign, Jlassign, Jassign

matches J global, local, or either assignment

Jlpar, Jrpar

match J's left and right parentheses

Jsol, Jeol

match the start or end of a J sentence  
(ignores leading blanks and trailing blanks/comments)

## Regular Expression Demo

This program demonstrates and allows you to experiment with regular expressions. Some standard text can be searched, or you can open any text file which will be displayed. When you type in a pattern and hit the Match button, the text will be searched for that pattern. All matches will be displayed in red and underscored.

A set of canned patterns can be tried by selecting them in the Patterns menu.

Run this demo from the **Studio|Demos** menu command. This demo requires a richeditm control and is only supported in the J Win95 and NT versions.

# Regular Expression Copyright

The J system and its interface to the regular expression pattern matching library is

Copyright (©) 1994-1998 Iverson Software Inc.

The regular expression library used in the interpreter is

Copyright (©) 1992, 1993, 1994, 1995 Free Software Foundation, Inc.

This library is being used according to the GNU Library Public License.

---

## GNU Library Public License

### GNU LIBRARY GENERAL PUBLIC LICENSE

Version 2, June 1991

Copyright (C) 1991 Free Software Foundation, Inc.

675 Mass Ave, Cambridge, MA 02139, USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

[This is the first released version of the library GPL. It is numbered 2 because it goes with version 2 of the ordinary GPL.]

### Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public Licenses are intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users.

This license, the Library General Public License, applies to some specially designated Free Software Foundation software, and to any other libraries whose authors decide to use it. You can use it for your libraries, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the library, or if you modify it.

For example, if you distribute copies of the library, whether gratis or for a fee, you must give the recipients all the rights that we gave you. You must make sure that they, too, receive or can get the source code. If you link a program with the library, you must provide complete object files to the recipients so that they can relink them with the library, after making changes to the library and recompiling it. And you must show them these terms so they know their rights.

Our method of protecting your rights has two steps: (1) copyright the library, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the library.

Also, for each distributor's protection, we want to make certain that everyone understands that there is no warranty for this free library. If the library is modified by someone else and passed on, we want its recipients to know that what they have is not the original version, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents.



We wish to avoid the danger that companies distributing free software will individually obtain patent licenses, thus in effect transforming the program into proprietary software. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

Most GNU software, including some libraries, is covered by the ordinary GNU General Public License, which was designed for utility programs. This license, the GNU Library General Public License, applies to certain designated libraries. This license is quite different from the ordinary one; be sure to read it in full, and don't assume that anything in it is the same as in the ordinary license.

The reason we have a separate public license for some libraries is that they blur the distinction we usually make between modifying or adding to a program and simply using it. Linking a program with a library, without changing the library, is in some sense simply using the library, and is analogous to running a utility program or application program. However, in a textual and legal sense, the linked executable is a combined work, a derivative of the original library, and the ordinary General Public License treats it as such.

Because of this blurred distinction, using the ordinary General Public License for libraries did not effectively promote software sharing, because most developers did not use the libraries. We concluded that weaker conditions might promote sharing better.

However, unrestricted linking of non-free programs would deprive the users of those programs of all benefit from the free status of the libraries themselves. This Library General Public License is intended to permit developers of non-free programs to use free libraries, while preserving your freedom as a user of such programs to change the free libraries that are incorporated in them. (We have not seen how to achieve this as regards changes in header files, but we have achieved it as regards changes in the actual functions of the Library.) The hope is that this will lead to faster development of free libraries.

The precise terms and conditions for copying, distribution and

modification follow. Pay close attention to the difference between a "work based on the library" and a "work that uses the library". The former contains code derived from the library, while the latter only works together with the library.

Note that it is possible for a library to be covered by the ordinary General Public License rather than by this special one.

## GNU LIBRARY GENERAL PUBLIC LICENSE TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License Agreement applies to any software library which contains a notice placed by the copyright holder or other authorized party saying it may be distributed under the terms of this Library General Public License (also called "this License"). Each licensee is addressed as "you".

A "library" means a collection of software functions and/or data prepared so as to be conveniently linked with application programs (which use some of those functions and data) to form executables.

The "Library", below, refers to any such software library or work which has been distributed under these terms. A "work based on the Library" means either the Library or any derivative work under copyright law: that is to say, a work containing the Library or a portion of it, either verbatim or with modifications and/or translated straightforwardly into another language. (Hereinafter, translation is included without limitation in the term "modification".)

"Source code" for a work means the preferred form of the work for making modifications to it. For a library, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the library.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running a program using the Library is not restricted, and output from such a program is covered only if its contents constitute a work based on the Library (independent of the use of the Library in a tool for writing it). Whether that is true depends on what the Library does and what the program that uses the Library does.

1. You may copy and distribute verbatim copies of the Library's complete source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and distribute a copy of this License along with the Library.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Library or any portion of it, thus forming a work based on the Library, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

- a) The modified work must itself be a software library.
- b) You must cause the files modified to carry prominent notices stating that you changed the files and the date of any change.
- c) You must cause the whole of the work to be licensed at no charge to all third parties under the terms of this License.
- d) If a facility in the modified Library refers to a function or a table of data to be supplied by an application program that uses the facility, other than as an argument passed when the facility is invoked, then you must make a good faith effort to ensure that, in the event an application does not supply such function or table, the facility still

operates, and performs whatever part of its purpose remains meaningful.

(For example, a function in a library to compute square roots has a purpose that is entirely well-defined independent of the application. Therefore, Subsection 2d requires that any application-supplied function or table used by this function must be optional: if the application does not supply it, the square root function must still compute square roots.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Library, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Library, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Library.

In addition, mere aggregation of another work not based on the Library with the Library (or with a work based on the Library) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may opt to apply the terms of the ordinary GNU General Public License instead of this License to a given copy of the Library. To do this, you must alter all the notices that refer to this License, so that they refer to the ordinary GNU General Public License, version 2, instead of to this License. (If a newer version than version 2 of the ordinary GNU General Public License has appeared, then you can specify that version instead if you wish.) Do not make any other change in these notices. Once this change is made in a given copy, it is

irreversible for that copy, so the ordinary GNU General Public License applies to all subsequent copies and derivative works made from that copy. This option is useful when you wish to copy part of the code of the Library into a program that is not a library.

4. You may copy and distribute the Library (or a portion or derivative of it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange.

If distribution of object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place satisfies the requirement to distribute the source code, even though third parties are not compelled to copy the source along with the object code.

5. A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a "work that uses the Library". Such a work, in isolation, is not a derivative work of the Library, and therefore falls outside the scope of this License.

However, linking a "work that uses the Library" with the Library creates an executable that is a derivative of the Library (because it contains portions of the Library), rather than a "work that uses the library". The executable is therefore covered by this License. Section 6 states terms for distribution of such executables.

When a "work that uses the Library" uses material from a header file that is part of the Library, the object code for the work may be a derivative work of the Library even though the source code is not. Whether this is true is especially significant if the work can be linked without the Library, or if the work is itself a library. The threshold for this to be true is not precisely defined by law.

If such an object file uses only numerical parameters, data structure layouts and accessors, and small macros and small inline functions (ten lines or less in length), then the use of the object file is unrestricted, regardless of whether it is legally a derivative work. (Executables containing this object code plus portions of the Library will still fall under Section 6.)

Otherwise, if the work is a derivative of the Library, you may distribute the object code for the work under the terms of Section 6. Any executables containing that work also fall under Section 6, whether or not they are linked directly with the Library itself.

6. As an exception to the Sections above, you may also compile or link a "work that uses the Library" with the Library to produce a work containing portions of the Library, and distribute that work under terms of your choice, provided that the terms permit modification of the work for the customer's own use and reverse engineering for debugging such modifications.

You must give prominent notice with each copy of the work that the Library is used in it and that the Library and its use are covered by this License. You must supply a copy of this License. If the work during execution displays copyright notices, you must include the copyright notice for the Library among them, as well as a reference directing the user to the copy of this License. Also, you must do one of these things:

a) Accompany the work with the complete corresponding machine-readable source code for the Library including whatever changes were used in the work (which must be distributed under Sections 1 and 2 above); and, if the work is an executable linked with the Library, with the complete machine-readable "work that uses the Library", as object code and/or source code, so that the user can modify the Library and then relink to produce a modified executable containing the modified Library. (It is understood that the user who changes the contents of definitions files in the Library will not necessarily be able to recompile the application to use the modified definitions.)

b) Accompany the work with a written offer, valid for at least three

years, to give the same user the materials specified in Subsection 6a, above, for a charge no more than the cost of performing this distribution.

c) If distribution of the work is made by offering access to copy from a designated place, offer equivalent access to copy the above specified materials from the same place.

d) Verify that the user has already received a copy of these materials or that you have already sent this user a copy.

For an executable, the required form of the "work that uses the Library" must include any data and utility programs needed for reproducing the executable from it. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

It may happen that this requirement contradicts the license restrictions of other proprietary libraries that do not normally accompany the operating system. Such a contradiction means you cannot use both them and the Library together in an executable that you distribute.

7. You may place library facilities that are a work based on the Library side-by-side in a single library together with other library facilities not covered by this License, and distribute such a combined library, provided that the separate distribution of the work based on the Library and of the other library facilities is otherwise permitted, and provided that you do these two things:

a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities. This must be distributed under the terms of the Sections above.

b) Give prominent notice with the combined library of the fact that

part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

8. You may not copy, modify, sublicense, link with, or distribute the Library except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, link with, or distribute the Library is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

9. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Library or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Library (or any work based on the Library), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Library or works based on it.

10. Each time you redistribute the Library (or any work based on the Library), the recipient automatically receives a license from the original licensor to copy, distribute, link with or modify the Library subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

11. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Library at all. For example, if a patent license would not per



## **Grid Control**

[Overview p162](#)

[Classes p163](#)

[Methods p164](#)

[Properties p165](#)

[Actions p166](#)

## Grid Control Overview

A grid is a display of data in cells. Grids are drawn on an isigraph control in a form.

Some wd commands are specific to grids and have names prefixed glgrid, for example, `glgridatt`; see the *Window Driver Command Reference*, gl2 commands. Apart from these wd commands, the grid is implemented in J.

The code that paints the grid is in an object derived from one or more [grid classes p163](#). The main grid classes included with J are:

jzgrid

basic grid class

jtgrid

table class, extends jzgrid.

Displays a function table in a grid.

jwgrid

watch class, extends jzgrid.

Displays a global variable in a grid.

A demo class, jfgrid (report class) is also included.

For an introduction to the grid control, see the labs *Grid Control*, *Grid Examples* and *Grid Low Level Programming*

## Classes

The basic grid class is `jzgrid`, and this provides facilities that should be required for any grid, including:

scrolling  
 mouse events  
 keyboard events  
 setting attributes, e.g. font, color, border, cell size

While the `jzgrid` class could in theory be used directly (for a very simple grid), there would typically be another class on top of the basic grid class that provides a specific type of grid, for example the watch class `jwgrid`. This would include data-related definitions such as providing formatted data for the grid and support for copying and editing.

As well as the grid classes, yet another class is needed for the Windows form definition. This will include

- the isigraph control to display the grid
- code to create the grid object from the grid classes to be used
- calls to grid event handlers

If you create a new form using File|New Class, the Form Editor Wizard can be used to add the appropriate definitions.

See Watch Example for a typical class structure.

## Watch Example

The following example using the watch grid illustrates the class paths:

```
dat=. ?30 30$100
```

This defines some data to be watched.

```
load 'jwatch'
```

This loads the class jwatch into the jwatch locale, which in turn loads the scripts jwgrid and jinput, each into their own locale. jwgrid in turn loads the basic grid script, jzgrid. At this stage, all that has happened is that these scripts have been loaded, and the locale namelist looks like:

```
conl ''
+-----+-----+-----+-----+
|base|j|jinput|jwatch|jwgrid|jzgrid|z|
+-----+-----+-----+-----+
```

No locale path settings have been made, so each locale has a path of z, as is the case with all new locales.

```
a=. conew 'jwatch'
```

This creates a new object (numbered locale) referenced by a. In a new session, it would be named '0' :

```
conl ''
++-----+-----+-----+-----+
|0|base|j|jinput|jwatch|jwgrid|jzgrid|z|
++-----+-----+-----+-----+
```

The new locale is empty, but has a locale path that includes jwatch and z.

```
create__a 'dat'
```

create\_\_a is called in the new locale, and gets its definition from jwatch. This verb creates a new form to display the grid, and then itself calls conew to create a jwgrid object and sets its path to include jwgrid and jzgrid:

```
grid=: ''conew'jwgrid'
```

Therefore the locale list now includes two numbered locales:

```

    conl ' '
+--+-----+-----+-----+-----+-----+--+
|0|1|base|j|jinput|jwatch|jwgrid|jzgrid|z|
+--+-----+-----+-----+-----+-----+--+

```

The paths in these two numbered locales are:

```

    copath a          locale '0'
+-----+--+
|jwatch|z|
+-----+--+

```

```

    copath grid__a    locale '1'
+-----+-----+--+
|jwgrid|jzgrid|z|
+-----+-----+--+

```

The structure of numbered locales is returned by `costate`:

```

    costate ' '
+-----+-----+-----+-----+-----+
|refs|id|creator|path          |
+-----+-----+-----+-----+
|a   |0 |base   |jwatch z      |
+-----+-----+-----+-----+
|    |1 |0      |jwgrid jzgrid z|
+-----+-----+-----+-----+

```

Now suppose the mouse is clicked on one of the grid cells. The form is defined in locale `a`, so the mouse event handler `watch_grid_mbldown` will be called there. Since this locale has no event handlers of its own, the definition in the `jwatch` locale is used. This is defined as:

```

    watch_grid_mbldown=: 3 : 'mbldown__grid sysdata'

```

Therefore the actual handler called is `mbldown` in the `grid` locale, which will pick

its definition up from either `jwgrid` or `jzgrid`, whichever it sees first.

## Method

The following are the public methods of the base jzgrid class.

## Method

### Description

create

resets grid globals

destroy

destroy object (cover for codestroy)

get

get data for grid, returns 1 if new data was required

gridselect

selects grid control (cover for glsel)

init

initialize grid. Argument is:

dataname name of grid data

gridid id of grid control

sbarid id of scrollbar

sbarvid id of scrollbarv

setdata

set new data in grid

setfont0

set font 0

setfont1

set font 1

setheights

set heights for range

setWidths

set widths for range

**Event handler methods** The following methods handle events for the grid control and are defined in jzgrid.

## **Method**

## **Description**

char

character event handler

mbldbl

mouse button left double click

mbldown

mouse button left down

mblup

mouse button left up



mbrdbl

mouse button right double click

mbrdown

mouse button right down

mbrup

mouse button right up

mmove

mouse button move

scrollbar

scroll horizontal bar

scrollbarv

scroll vertical bar

size

size event handler

**Watch Methods** The following are the public methods of the watch class jwgrid. In some cases, the watch class method overrides the base method.

## **Method**

## **Description**

copy

copy selected cells to clipboard

create

resets grid globals

destroy

destroy method

paste

paste clipboard into grid

setdata

set new data in grid

sethighlight

show highlights in grid

## Properties

The following are the public properties of the base jzgrid class.

### Name

### Default

### Description

atts

see resetatts

attribute matrix

borders

{ empty }

border definitions

dataname

{ none }

name of variable displayed in grid

defheight

20

default cell height

defwidth

50

default cell width

editenable

0

set to 1 when editing is enabled

editflag

0

set to 1 if editing cell

extent

1 1

extent of selection

filltype

0

fill type for other cells

fixc

1

number of fixed columns

fixeddata

0

1 if all data is supplied at one go

fixr

1

number of fixed rows

fixtype

0

fill type for a fixed cell

font0

courier new 12

font0, default "'courier new" 18'

font1

arial 12

font1, default 'arial 18'

formhwnd

{none}

form window handle

gridhs

{empty}

grid heights

gridhwnd

{none}

grid control handle

gridws

{empty}

grid widths

mark

0 0

top left row,col of selected data

maxc

—

max number of columns

maxr

—

max number of rows

sbarid

{none}

id of the scrollbar

sbarvid

{none}

id of the scrollbarv

scrollc

0

cols scrolled out of view

scrollr

0

rows scrolled out of view

sizeenable

0

set to 1 if user can resize cells

skipc

0

skipped cols

skipr

0

skipped rows

The following are the public properties of the jwgrid watch class.

**Name**

**Description**

highlight

highlight expression

## **Actions**

The following actions are defined in jzgrid.

### **Action**

### **Description**

arrows

move selection (scrolls to keep visible)

shift+arrows

extends range

ctrl+arrows

scrolls data

ctrl+shift+arrows

changes default cell size

home

scrolls left to col 0

ctrl+home

scrolls to row 0 and col 0

ascii characters

new data for selected cell

enter



end edit mode, move to next row

alt-enter

edit mode new line

click

selects cell

double-click

set edit mode on for current cell data, if edit is enabled

mouse-down & hold

extends range of selection to right and down

ctrl+e

enable/disable edit

ctrl+f

edit font

A specific grid may define additional or alternative behavior. For example, the watch grid adds:

ctrl+m

enter highlight expression

## **Plot**

[Overview p168](#)

[pd verb p169](#)

[plot verb p170](#)

[Plot Class p171](#)

[Plot Types p172](#)

[Plot Commands p173](#)

[Plot Options p174](#)

[Plot Data p175](#)

[Plot Colors p176](#)

## Plot Overview

Plot is a plotting package that provides business and scientific graphics.

Plots can be drawn directly from the J session, or included in a Windows form.

For example:

```
load 'plot'  
plot */~ i.20
```

This loads the plot system and draws a simple plot.

There are two user verbs, `pd` (plot driver) and `plot`.

- `pd` is the low-level verb that handles all calls to Plot.
- `plot` is a cover verb for `pd` that will handle most simple uses of Plot.

For an introduction, see the lab *Plot Package*.

For examples illustrating the main facilities provided, see the demo *Studio/Demos.../plot*. In the demo, use *Options/View Definition* to view and experiment with the plot definitions.

To use the plot class, for example, to include plots on a Windows form, see [Plot Class p171](#)

## pd verb

pd is the low-level verb that handles all calls to Plot.

pd has three types of argument:

## commands

set up and show a plot

[Plot Commands p173](#)

## options

specify plot type, colors etc

[Plot Options p174](#)

## data

data used in the plot

[Plot Data p175](#)

Commands and options are given in a character list, delimited by semicolons. Data is a numeric or boxed array.

For example:

load 'plot numeric trig'	NB. load plot and some utilities
pd 'reset'	NB. command reset
pd 'textc 500 950 My Plot'	NB. command textc
pd 'type line'	NB. option type
pd cos steps 0 8 100	NB. data
pd 'show'	NB. command show
pd 'clip'	NB. command clip
pd 'print'	NB. command print

## Plot Verb

`plot` is a cover for `pd`.

The optional left argument is a list of commands and options, delimited by semicolons. The right argument is data.

`plot` and `pd` may be used together, for example:

<code>'type bar' plot &gt;:?2 6\$10</code>	NB. draw a barchart
<code>pd 'print'</code>	NB. print it

## Plot Class

Plot is defined in class `jzplot`.

Public verbs are `pd` and `plot`.

Public nouns specify the Windows form and isigraph control id, where these differ from the default plot.

### Name

### Description

### Default

`PForm`

form handle

`plot`

`PFormhwnd`

form handle

defined when the form is created

`PIid`

isigraph control id

`gs`

### Example: create two plots

The following draws two independent plots:

```
load 'jzplot'          NB. load plot class
```

```

a=: conew 'jzplot'      NB. create plot object a
b=. conew 'jzplot'      NB. create plot object b
plot__a */~ i.20        NB. draw plot in a

```

```

plot__b +./~ i.20      NB. draw plot in b

```

## Example: add plot to a form

In this case, you need to define the public nouns PForm, PFormhwnd and PId. These override the default values.

```

load 'jzplot'

```

```

MYPLOT=: 0 : 0
pc myplot closeok;
xywh 2 2 125 100;cc g0 isigraph ws_border rightscale bottommove;
xywh 131 3 34 11;cc close button leftmove rightmove;cn "Close";
pas 2 2;pcenter;
rem form end;
)

```

```

myplot_close_button=: wd bind 'pclose'

```

```

wd MYPLOT

```

```

a=: conew 'jzplot'      NB. create plot object
PForm__a=: 'myplot'     NB. define PForm in a
PFormhwnd__a=: wd 'ghwndp' NB. define PFormhwnd in a
PId__a=: 'g0'           NB. define PId in a

'density' plot__a 7|i.25 25  NB. draw plot on the form

```

## Plot Types

### Type

**2D**

**3D**

### Description

area

X

shows area under one or more lines

bar

X

bar chart

density

X

density plot (2D representation of 3D data)



dot

x

dot plot

errorbar

x

error bars

hilo

x

hi - lo plot

line

x

x

line plot (3D if x;y;z values given)

linefit

x

fitted line plot

fbar

x

floating bar chart

sbar

x

step bar chart

pie

x

pie char

point

x

point plot

radar

x

radar plot

stick

x

x

stick plot (vertical bars)

step

x

step plot

surface

x

surface plot

wire

x

wire surface plot



## Plot Commands

The units used in these commands are based on an isigraph window of 1000 by 1000. The point 0,0 is the bottom left corner.

Some of these commands refer to "gl" commands which are documented in the wd Commands online help.

### Command

### Description

#### **reset [parent]**

reset plot with optional parent window id.

For example:

```
pd 'reset'
```

#### **new [window]**

starts new plot with optional window, sets user defaults. Use to set initial values, and when display multiple plots. The default window is 0 0 1000 1000.

For example, create a new plot window with xy position 100 100, and of size 400 by 600:

```
pd 'new 100 100 400 600'
```

#### **use window**

change window

**text, textc, textr arg**

text (left aligned), centered, right aligned.

Argument is x y text, where x y are coordinates relative to window.

For example, the following writes the text "J Graphics" centered at position 500 950:

```
pd 'textc 500 950 J Graphics'
```

**{options}**

sets plot options. See Plot Options

For example:

```
pd 'backcolor white; frame 1'
```

**{numeric data}**

sets plot data. More than one set of data can be given.

**clip [w h]**

copy plot to clipboard in wmf format

w h are clipboard width and height in 0.01 millimeters, default 10000 7500

**show**

displays plot

## **save [file w h]**

save plot to file in wmf format

file is filename. The file extension defaults to .wmf. If file not given, 'save' is treated as 'clip'.

w h are relative width and height. Default is current plot width and height. For example,

1000 1000 same width and height

1000 500 width is twice height

**savebmp [file w h]** save plot as 24bit color bitmap file.

file is filename. The file extension defaults to .bmp. If file not given, the bitmap is copied to the clipboard.

w h are width and height in pixels, default 512 512.

## Plot Options

Plot options and their defaults are shown below. The defaults are set whenever you use the 'reset' or 'new' commands and are defined in script `system\classes\plot\plotdefs.ijs`.

Options are specified by providing the option name followed by its argument. Several options can be given at one time, separated by semicolons. Option values are stored as uppercase global variables.

Options whose names begin with 'x' apply to the x-axis; corresponding options apply to the y and z axes.

"Isigraph units" are based on graphics window size of 1000 by 1000.

Some of these options refer to "gl" commands which are documented in the `wd` Commands online help.

## Options

**Type**

**Default**

## Description

**aspect**



2r3

ratio of graphics window height to width. Used to adjust ticmark lengths, and graph boxes for a pie chart.

**axes**

b

1 1

if x,y axes are shown

**backcolor**

color

WHITE

background color

**bandcolor**

colors

see Plot Colors

band color scheme

**border**

b

0

if a border is drawn

**bordersize**

n

8

size of border (isigraph units)

**boxed**

b

1

if drawn in a box (3D only)

**captionfont**

font

Arial 40

font for x and y captions

**clear**

b

1

if clear the background before drawing plot

**color**

colors

see Plot Colors

plot colors

**edgecolor**

colors

BLACK

edge color (ellipse, pie, poly, rect)

**forecolor**

color

BLACK

foreground color (for axes, text)

**frame**

b

0

if plot is framed

**gridcolor**

color

GRAY

grid line color

**grids**

b

0 0

if x, y grids are shown

**itemcolor**

colors

see Plot Colors

item color scheme

**key**

c

{none}

key names (legend identifying plot items)

**keyfont**

font

Courier New 25

font for key names

**keystyle**

n

0 2

0=horizontal 1=vertical style 0-3 positions from top left

**labelfont**

font

Courier New 40

font for labels

**labels**

b

1 1

if x,y labels shown

**mesh**

b

1

if a mesh is drawn (3D only)

## **orientation**

n

1

1=portrait, 2=landscape (printing only)

## **penpattern**

v

see plotdefs

patterns used by pen styles

## **pensize**

n

1

pensize (see definition of glpen)



**penstyle**

n

solid

pen style (see definition of glpen)

**plotbox**

n

0 0 1000 1000

position and size of plotting window.

**plotcaption**

c

Plot

plot caption

**polar**

b

0

if a polar plot, data is r;theta (2D only)

**rtic**

v

{none}

r tics (#major,#minor) (radar plot only)

**separator**

c

|

separator for char matrix entries

**symbolfont**

font

Symbol 40 bold

symbol font used in point plots

**textcolor**

color

BLACK

text color

**textfont**

c

Arial 40

text font

**ticmajor**

v

12

size of major tic marks (isigraph units)

**ticminor**

v

8

size of minor tic marks (isigraph units)

**tics**

v

1 1

if x, y tic marks shown

**title**

c

{none}

title text

**titlefont**

c

Arial 60

title font

**type**

c

line

plot type (see Plot Types)

**viewcenter**

v

0 0 0

position of plot center (3D only)

**viewpoint**

v

1.6 \_2.4 1.5

position of observer (3D only)

**viewsize**

v

1 1 0.5

relative sizes of viewbox

**viewup**

v

0 0 1

upwards direction (3D only)

**visible**

b

1

if plot is displayed

**xcaption**

c

{none}

x caption

**xgridpattern**

v

3 5

x grid pattern

**xint**

n

{none}

position of x-intercept

**xlabel**

c

{none}

x labels

**xlog**

n

0



if log applied to x values

**xrange**

v

{none}

range of x data - low,high

**xtic**

v

{none}

x tic (style)

**xticpos**

v

{none}

x tic positions

## **Type**

## **Description**

**b**

boolean 0 or 1

**c**

character string

**color[s]**

color or color matrix - given by name or values

**font**

font specification

**n**

number

**v**

numeric list

## Plot Data

Plot data may be given as either a numeric or boxed array.

A numeric array should be a vector or matrix.

- For a 2D plot, a vector is the y values and a matrix is treated as rows of y values. The x axis defaults to  $i . \{ : \$Y$  .
- For a 3D plot, the array should be a matrix and is treated as z values. The x axis defaults to  $i . \# \ z$  and the y axis to  $i . \{ : \$z$  .

A boxed array is either x;y values for a 2D plot, or x;y;z values for a 3D plot. The boxed values should conform in size.

- For a 2D plot, the x values should have the same shape as the y values, or be a vector of length  $\{ : \$Y$  .
- For a 3D plot, the x and y values should have the same shape as the z values, or the x values should have length  $\#z$  and the y values of length  $\{ : \$z$  .

## Plot Colors

Several color schemes are used in Plot.

Before any drawing is made, the *backcolor* is applied to the plot box. The axes, frame, ticmarks, labels and titles are then drawn in the *forecolor*.

Text specified with the `text/textc/textr` commands is drawn with the *textcolor*.

Data is colored in two ways, depending on plot type:

- where each data item is plotted in a single color (e.g. line plots), the items are drawn with the *itemcolor*
- where a data item is banded and requires several colors to plot, (e.g. density plots), it is drawn with the *bandcolor*.

The edges of filled-in shapes are drawn with the *edgecolor* (usually black).

Backcolor, forecolor, textcolor and edgecolor are single colors.

Itemcolor and bandcolor are lists of colors. Itemcolor typically contains quite distinctive colors to distinguish the different data items. Bandcolor is typically a graduated scale of colors, for example to indicate height. Typical lists of colors are defined in `system\classes\plot\plotdefs.ijs`.

The term *color* can also be used as an abbreviation to specify the itemcolor or the bandcolor depending on plot type. This convenient for simple plots where only one plot type is being drawn.

For example:

```
pd 'textcolor red'
```

single color for text

```
pd 'itemcolor blue,red,green'
```

list of colors for items

```
pd 'bandcolor bgclr'
```

list of colors in color band

```
pd 'color blue,red'
```

item or band colors (depends on plot type) The script  
system\classes\plot\plotdefs.ijs defines several color schemes, for example:

- STDCLR is the color scheme used by default
- RBCLR is a red/blue color scheme that is appropriate for 3D surface plots

## **OpenGL**

[Movement Keys p178](#)

[Viewing p179](#)

## Movement Keys

The following keys are active when an OpenGL graphic is displayed and has the current focus.

Keys	Movement
i o	in / out (shift = 5 steps at a time)
x y z	Rotate about axis (shift = rotate back)
F5 F6 F7	Rotate about axis by 45_ (shift = rotate back)
Ctrl F5 F6 F7	Rotate about axis by 180_
j k l	Move object in x, y, z direction (shift = move back)
up/down/left/right	Move view position (fly around)
PageUp/Page Down	Spin view position
1-9	Change speed (1=slowest, 5=default, 9=fastest)
F3	Snap current state
F4	Return to snap state or initial state if none.
Shift F4	Return to initial state.



## Viewing

Graphics are drawn at the origin, and then three viewing controls are applied. First, the graphic is rotated about one or more axes. Next, it is translated (moved) to a new position. Finally, the observer's view position is given.

These viewing controls are always applied. If not specified, defaults are used. Rotation and translation may be zero, leaving the graphic at the origin, as originally drawn.

(Since OpenGL applies viewing commands in the reverse order in which they are specified, they are here specified in the order: view, translate, rotate.)

## Rotation

The rotation angles are given in degrees in ROTXYZ, and are applied in the order ROTNDX.

By default ROTNDX is 0 1 2, meaning the specification order is x-axis, y-axis and z-axis (and applied by OpenGL in the order z, y, x). When setting up a graphic, it is easiest to leave ROTNDX as the default. However, the paint handlers may change the order to ensure smooth rotations when using the x y z keys to rotate the graphic.

## Translation

Translation moves the graphic away from the origin, by the amount in TRNXYZ. In most cases, there is no need to move the graphic.

## Viewing

The position of the observer is given in VIEWXYZ, and the up direction in VIEWUP. The observer always looks at the origin. If you set VIEWXYZ manually, ensure that VIEWUP is also given a suitable value.

## Example

A good way to experiment with viewing is to use the OpenGL demo

Basic/Viewing. This shows the viewbox with the positive axes. Try to guess which way the graphic will move when each key is pressed.

## **Java**

[Java p181](#)

[Java jserver class p182](#)

[Java classpath p183](#)

[Jsoftware Java applets p184](#)

[Java examples p185](#)

[Java applet security p186](#)

## Java

J Automation objects JEXEServer and JDLLServer are available to Java programmers. A Java programmer can easily use the power of J in building applications and applets.

J Automation objects and the corresponding Java support are only available in the Windows 95 and NT versions of J.

J User Manual chapter OLE gives an overview of J Automation objects. The term Automation supersedes the term OLE.

Automation is a Microsoft technology and it is supported directly in the MS J++ Java Development system, the MS Java Virtual Machine (JVM) jview.exe, and in Internet Explorer.

There are two problems with this direct MS support of Automation. JVMs from other vendors (e.g., Sun's java.exe) and Netscape browsers can't run Java programs that use Automation objects. There are also serious limitations in mapping data between J and Java with the MS Automation support.

These problems were solved by building a Java class that wraps the J Automation objects in a way that is identically usable in all JVMs and browsers. This Java class is the jserver class and it is distributed with the J system.

The best way to learn how to use J in Java is to study the examples. Run them, build them in your Java development environment, and experiment. Stepping through with a debugger is a good way to study the examples.

## Java jserver class

The jserver class is a wrapper for the J Automation objects JEXEServer and JDLLServer. The jserver.class is in package isi.ijservice (Iverson Software Inc.) and is distributed with the J system. To use a J object you import isi.ijservice into a Java source file and use the class methods and fields.

The jserver class provides identical access for all JVMs to J Automation objects and it provides methods that simplify mapping J data to Java. The jserver class uses native methods in a C++ dll. There are two versions of the native method dll: jsnmms.dll (J Server Native Method MS) and jsnmns.dll (Netscape and other JVMs).

The jserver class uses native methods and must exist in the local file system. The class and native method dlls must be in the J system\extras\java\classes\isi\jserver directory. This path is added to the Java classpath environment variable when J is installed.

```
// constants - results of getType()
final static int TYPEBOOL=1;
final static int TYPEBYTE=2;
final static int TYPEINT=4;
final static int TYPEDOUBLE=8;
final static int TYPEBOX=32;

// methods
synchronized void start(int dllv) // 0 for JEXEServer, 1 for JDLLServer
int close() // close J object
int Do(String s)
int DoR(String s) // Do with formatted result
int Show(boolean b)
int Log(boolean b)

// methods for getting J data
int Get(String s) // get data for J variable

int getType()
```

```
int getRank()  
int getElements()  
int[] getShape()
```

```
boolean getBool()  
char getChar()  
int getInt()  
double getDouble()
```

```
boolean[] getBools()  
byte[] getBytes()  
int[] getInts()  
double[] getDoubles()  
String getString()
```

```
// methods for setting J data  
int Set(String s, Object x)
```

```
int Set(String s, byte x)  
int Set(String s, int x)  
int Set(String s, double x)
```

## **Java classpath environment variable**

A Java class is a .class file that contains the compiled result of java source files. When a JVM runs a program it dynamically loads the java classes required. When the jserver.class is required it must be loaded from your local file system. Different JVMs have different search paths for class files, but they all search the classpath path. Rather than add the jserver class file to different locations, the J installation updates the classpath environment variable to include the J system\extras\java\classes directory.

## **Java [www.jsoftware.com](http://www.jsoftware.com) applets**

[www.jsoftware.com](http://www.jsoftware.com) has a few simple applets that you can run in your browser. When the applet uses the jserver class, it is loaded from the J system\extras\java\classes directory.

The nyse applet is different from the others in an important way because it reads data from a URL at the NYSE web site. Applets downloaded from the web run in the Java sandbox and are not allowed to read data from other web sites. The nyse applet would get a security violation if it were downloaded from the web site. The nyse applet class is in your J directory system\extras\java\classes. If this is in your classpath environment variable (as set by J setup) then the class is loaded from your hard drive and is not restricted to running in the sandbox. Installing applet classes on your hard drive and adding paths to your classpath have serious security implications. Be sure to read: [Java sandbox and applet security p186](#).



## Java examples

The source for several Java programs is included in J directory system\examples\java. The examples are simple and could be understood just by reading, but it is helpful to use a debugging environment to step through the code to see exactly what happens.

Most of the examples are applets. Directory jstest contains an application. The jstest.class file is in the J system\extras\java\classes directory. You can run jstest with the MS JVM jview.exe or the Sun JVM java.exe. Bat file jv.bat runs jview.exe and ja.bat runs java.exe.

## Java sandbox and applet security

A java applet downloaded from the web runs on your system in a secure environment called the sandbox. An applet running in the sandbox is not allowed to do anything that could violate the security of your system. In particular it can't read or write files. It is very important that a J object created in the sandbox obeys the rules. J foreigners ( ! : ) and wd commands that could violate sandbox rules are disabled when the J object is created in the sandbox.

## Important exception: loading ijs script files is allowed.

Be very careful about adding applet classes to your hard drive or changing your classpath. An applet loaded from your hard drive has complete access to your system. Allowing a strange applet to run from your hard drive is just as dangerous as running a strange exe program on your system.

It might sometimes be useful to explicitly request sandbox security completely independently from Java and Automation. For example, you might want sandbox security to run a script downloaded from the net. You trust the script, but setting sandbox security makes it safer. You can explicitly set sandbox security:

```
wd'security 1'      NB. wd sandbox security
9!::24 [ 1         NB. !: sandbox security
```

## **Sockets**

[Socket Driver p188](#)

[Socket Utilities p189](#)

## Socket Driver

A socket is an endpoint in a bi-directional communication channel. The other end can be in the same task (not usually very interesting), in another task on the same machine, or in a task on another machine that is accessed through a TCP/IP connection.

The foreign family 16! : support sockets under NT and Win95. The Socket Driver is not available on the Macintosh. The Socket Driver works with 32 bit TCP/IP and does not work with a 16 bit TCP/IP stack.

File system\main\socket.ijs contains cover functions for the new Socket Driver. To load, enter:

```
load 'socket'
```

Directory system\examples\socket contains examples of using the Socket Driver to communicate between two J sessions. See system\examples\socket\socket.txt for details.

To use sockets you need to have TCP/IP support configured. The following is an excerpt from a FAQ:

[Q: How do I set up my computer for a TCP/IP network?]

1. In Control Panel, double-click the Network icon.
2. On the Configuration tab, click Add, and then double-click Protocol.
3. Click Microsoft, and then click TCP/IP

After it is installed, click TCP/IP on the Configuration tab of Network properties, and then click Properties. Configure your protocol per instructions from your system administrator.

The Socket Driver is a very direct mapping onto the Windows Sockets 1.1 API interface. General documentation on sockets and the Windows API is relevant and useful for complete understanding of how best to make use of sockets. Web sites [www.stardust.com](http://www.stardust.com) and [www.sockets.com](http://www.sockets.com) have lots of relevant information for a

serious socket application developer. In particular, you might want to download the complete Windows Sockets 1.1 Specification from:  
[http://www.stardust.com/winsock/ws1.1\\_api](http://www.stardust.com/winsock/ws1.1_api)

There are labs (Studio|Labs) on sockets.

## Socket Utilities

Script `system\main\socket.ijs` has definitions for working with sockets.

The first element of the result of all socket verbs is a result code. It is 0 if there was no error, or it is an error number. Utility `sderror` returns the text name of the error number.

Some socket verbs take integer constants as arguments and some of these constants have been given names in `socket.ijs`. For example `SOCK_STREAM` is 1.

A socket can be blocking or non-blocking. A verb such as `sdrecv` will hang on a blocking socket until it can complete. On a non-blocking socket, the `sdrecv` returns immediately with a result code of `EWOULDBLOCK 10035`. In Windows the use of non-blocking sockets is recommend.

A socket created with `sdsocket` is a blocking socket. The verb `sdioctl` can make it non-blocking. In Windows it is recommended to use `sdasync` to make the socket non-blocking as this also marks the socket so that events are notified to J by running sentence `socket_handler''`. This is similar to window events and the `wdhandler''` sentence.

Verb `sdselect` returns information about the state of a socket and indicates if it has data to read, is ready for writing, or has had an error.

Addresses used with sockets consist of 3 parts: family, address, port. The first is an integer which indicates the type of address. Currently this is always `AF_INET` (address family internet). The second part is a string that is a series of 1 to 4 numbers separated by dots. The third part is an integer port.

```
sdsocket  family , type , protocol
sdsocket  ''
```

Creates a new socket. A socket is identified by an integer. The family must be `AF_INET` as defined in `sockets.ijs`. The type can be any of the `SOCK_` values, but is usually `SOCK_STREAM`. The protocol must be 0. The result is a socket number that can be used as the socket argument to other verbs. The " argument is equivalent to

```
AF_INET,SOCK_STREAM,0sdrecv  socket ,  
count , flags
```

Receives data from a socket. The count is the maximum amount of data that will be received. The flags are from the `MSG_` values and is usually 0. The result is a boxed list. The first element is the result code and the second is the data. There may be less data received than in count.

If the socket is blocking and there is no data, the verb will hang until there is data.

If the socket is non-blocking and there is no data, it immediately returns result code `EWOULDBLOCK 10035`.

`sdioctl` can be used to see how much data is available for a socket.

If the socket at the other end is closed, then the socket will be in the `sdselect` ready-to-read list and an `sdrecv` will immediately receive 0 characters with no error.

If `sdasync` has been done for a socket, then the `socket_handler''` is run whenever new data is available.  
`sdrecv sk , 1000 , 0`

```
data sdsend socket , flags
```

The left argument is the data to send. The flags are from the `MSG_` values and is usually 0.

Blocking and non-blocking sockets work with `sdsend` in a similar manner to `sdrecv`.

The second element of the result indicates how many characters were actually sent. This may be less than was requested and you need to call `sdsend` again to send the

remaining data. 'testing 1 2 3' sdsend sk , 0  
sdrecvfrom socket , count , flags

Similar to sdrecv except it is used with a SOCK\_DGRAM socket. The result has additional elements that give the address of the data source.

data sdsendto socket ; flags ; family ; address ; port

Similar to sdsend except it is typically used with a SOCK\_DGRAM socket and the argument includes the address to send the data to. 'test' sdsend sk ; 0 ;

AF\_INET ; '127.0.0.1' ; 800

sdclose socket

Close a socket.

sdconnect socket , family , address , port

Connect the socket to the socket indicated by the address.

An sdconnect on a blocking socket will hang until it completes and will either return a 0 result code indicating success, or an error code.

An sdconnect on a non-blocking socket returns immediately with EWOULDBLOCK 10035. The system will try to complete the connection asynchronously. If it is successful, the socket will be marked ready for writing in sdselect. If the connection fails the socket will be marked in error in sdselect. sdconnect sk ;

AF\_INET ; '127.0.0.1' ; 800

sdbind socket , family , address , port

Bind a socket to an address. The address can be " if the socket will be used to listen for connects to any address on the machine. If the port number is 0, the system will assign a port (which can be queried with sdgetsockname).

A bind is usually done with a socket that will listen for connections. sdbind sk ;

AF\_INET ; '' ; 800 NB. any connections to 800

sdlisten socket , number

Set the socket to listen for connections. A bind must have been done. The number is the limit to queued connections. The host typically forces this limit to be between 1 and 5.

When a connection is made the socket is marked in sdselect as ready for reading. When it is ready sdaccept should be done.

sdaccept socket

When a listening socket is marked as ready for reading in sdselect, then an accept can be done to create a new socket for this end of the channel. The new



socket is a clone of the listening socket and has all its attributes. In particular, if the listening socket is non-blocking or has been marked with `sdasync`, then the new socket is as well. The result is the result code and the new socket.

```
sdselect  read ; write ; error ; timeout
sdselect  ''
```

The argument is a 4 element list. The first is a list of sockets to check for ready-to-read, the second is a list to check for ready-to-write, and the third is a list to check for errors. The last element is a timeout value in milliseconds. If it is 0, the select is non-blocking and returns immediately. If the timeout is not 0, it will return as soon as there is a socket to report on, but will not wait longer than the timeout value.

An empty argument checks all sockets for all conditions with a timeout of 0.

The result has a result code and 3 socket lists. The first is the list of ready-to-read sockets. The second is a list of ready-to-write sockets. The last is a list of sockets that had an error.

Ready-to-read sockets are sockets with data available for an `sdrecv` or listening sockets with an incoming connection

```
sdgetsockopt  socket , option_level , option_name
```

Returns the value of a socket option.  
`sdgetsockopt sk , SOL_SOCKET , SO_DEBUG`

```
sdgetsockopt sk , SOL_SOCKET , SO_LINGER
```

```
sdsetsockopt  socket , option_level, option_name , value...
```

Set the value of a socket option.

```
sdsetsockopt sk , SOL_SOCKET , SO_DEBUG , 1
```

```
sdsetsockopt sk , SOL_SOCKET , SO_LINGER , 1 , 66
```

```
sdioctl  socket , option , value
```

Read or write socket control information.

```
sdioctl sk , FIONBIO , 0  NB. set blocking
```

```
sdioctl sk , FIONBIO , 1  NB. set non-blocking
```

```
sdioctl sk , FIONREAD, 0  NB. count of data ready to read
```

```
sdgethostname  ''
```

Returns host name.

```
sdgetpeername  socket
```

Returns address of socket this socket is connected to.

```
sdgetsockname  socket
```

Return address of this socket.

```
sdgethostbyname name
```

Returns an address from a name.

```
sdgethostbyname 'localhost'
+--+-----+
|0|2|127.0.0.1|
+--+-----+
sdgethostbyname >1{sdgethostname ''
+--+-----+
|0|2|204.92.48.126|
+--+-----+
sdgethostbyname 'www.jssoftware.com'
+--+-----+
|0|2|198.53.145.167|
+--+-----+
sdgethostbyaddr AF_INET , address_name
```

Returns a name from an address.

```
sdgethostbyaddr 2 ; '127.0.0.1'
+--+-----+
|0|localhost|
+--+-----+
```

```
sdgetsockets ''
```

Return result code and all socket numbers.

```
sdwsaasync socket
```

Make a socket non-blocking and cause the system to run sentence socket\_handler" whenever the state of the socket has changed.

```
sdcleanup ''
```

Close all sockets and release all socket resources.

## J Engine Protocol

Jconsole in both Windows and Unix provides a console interface to the J Engine (JE). Jconsole can also be started with a command line parameter so that it instead provides a socket interface to the JE that uses the J Engine Protocol. The client side of this interface to the JE server is a J Front End (JFE).

JFE starts JE with a command line requesting it to connect to a socket. JFE can start JE on the same machine, in which case it needs to provide a port number. Or it can start JE on another machine, in which case it also needs to provide the host name or address. For example:

```
jconsole -jconnect 1234  
jconsole -jconnect frodo:2002  
jconsole -jconnect 192.168.1.3:4321
```

The J Socket Protocol allows any client with sockets to have full use of a JE. This provides facilities that are similar to J OLE Automation in Windows, but does so in a portable, open, more efficient, and much simpler manner.

Jsoftware provides a JFE written in Java that runs in both Windows and Unix. This JFE supports the J wd interface that provides GUI facilities to the J programmer and provides a standard, portable, cross-platform IDE (Interactive Development Environment).

Jsoftware also provides a script (open'jserver') that lets J act as a JFE to another JE. Studying and playing with this script is a good way to learn about the J Socket Protocol. See the jserver script for details like the values for names (e.g. CMDDO) used in this document.

## Messages

The JFE client and the JE server exchange messages over the socket. Some messages require a reply and others don't.

All messages have the following format:

Command - 1 byte

reserved - 3 bytes

Type - 4 byte integer (NBO) with extra command info

Len - 4 byte integer count (NBO) of bytes in data

Data - Len bytes of data

In NBO (network byte order or standard byte order) the bytes 0 0 0 1 are the integer 1. In RBO (reverse byte order) the bytes 1 0 0 0 are the integer 1. In HBO (host byte order) the byte order is NBO or RBO depending on the host. Intel x86 and related machines are RBO and most others are NBO.

Len can be 0, in which case there are 0 Data bytes. Data bytes are not null terminated and can have any value.

### Command Summary

Command	Sent by	Type	Data	Description
CMDDO	client	0	string	JE executes sentence
CMDDOZ	JE	error	none	JE finished with sentence
CMDIN	JE	0	prompt	J requests keyboard input
CMDINZ	client	0	input	JE resumes execution
CMDWD	JE	x	array	client runs 11!:x request
CMDWDZ	client	WDZ	data	JE resumes execution
CMDOUT	JE	OUT	string	output for display
CMDBRK	client	0	none	JE interrupts execution
CMDGET	client	0	name	JE replies with CMDGETZ
CMDGETZ	JE	error	array	value in 3!:1 format
CMDSETN	client	0	name	sets name for CMDSET
CMDSET	client	0	array	sets name with 3!:1 value
CMDSETZ	JE	error	none	result of set
CMDED	client	0	event	null terminated strings for wd'q'
CMDFL	JE	0	locale	locale for next wd commands

CMDEXIT JE code none 2!:55 code

Type in CMDWDZ describes the Data format:

WDZSTR - string

WDZINT - list of integers in JE HBO

WDZMTM - Len is 0 and the J result is an empty matrix

WDZERR - Len is 0 and Type is 1000+error (e.g. 1003 is domain error)

Type in CMDOUT describes the output:

OUTFR - formatted result array (just before CMDDOZ)

OUTERR - error output

OUTLOG - output from loading script

OUTSYS - system assertion failure

OUTFILE - 1!:2[2

error Type is 0 for no error or a J error (e.g. 3 is domain error).

Data array is the 3!:1 binary representation of a J array.

CMDED (Event Data) sends the data for an event to JE before a CMDDO with the wdhandler sentence. JE gets this event data with wd'q' which runs locally in the JE.

CMDFL (Form Locale) sends the current locale to JFE before a CMDWD for 11!:0. The locale name is required by the client so it can be included in an event for a form created by a pc command.

## **Command Transactions**

Some messages require a response.

DO/DOZ - client sends a sentence and JE replies when finished

WD/WDZ - JE sends a wd command and the client replies when finished

GET/GETZ - client sends a name and JE replies with its 3!:0 value

SETN/SET/SETZ - client sends name, then 3!:0 value, and JE replies when finished

DO/DOZ is the main transaction. IN/INZ/WD/WDZ/OUT/BRK/ED/FL messages occur only only within a DO/DOZ transaction.

DO, GET, SETN, and SET commands can only be sent when J is ready (i.e., not already working on a message).

### **Command Sequences**

sentence with no output:

client DO a=.5

server DOZ

sentence with output:

client DO i.5

server OUT OUTFR 0 1 2 3 4

server DOZ

1!:2[2 (smoutput) output

client DO i.5[smoutput 'test'

server OUT OUTFILE test

server OUT OUTFR 0 1 2 3 4

server DOZ

error:

client DO 'a'+1

server OUT OUTERR domain error

server DOZ

1!:1[1 request for input (or a debug suspension):

client DO ...

server IN prompt

client INZ input

server OUT OUTFR formatted result

server DOZ

break:

client DO 6!:3[100 NB. long running sentence

client BRK

server OUT OUTERR interrupt

server DOZ

wd command:  
client DO wd'pc abc;cc b button;pshow'  
server FL base  
server WD 'pc abc;cc b button;pshow'  
client WZ WZMTM  
server OUT OUTFR formatted result  
server DOZ

wd event:  
client ED event data  
client DO wdhandler\_formlocale\_  
... server DOZ

## Array Data

Array data is the 3!:1 representation of a J array. Numerics are in the JE HBO except for SET which can be either NBO or RBO.

The array format is: type, flag, count, rank, shape, data

The type, flag, count, rank, shape are 4 byte integers. The type is defined as in 3!:0. The flag must be 0. Boxed arrays are nested with values as offsets to the array.

You can experiment in J to see the bytes in the data array. Try the following sentences.

```
hex=: 3!:3 NB. hex display of 3!:1 - host byte order
hex 'abc'
hex 'abcd'
hex 2 2$'abcd'
hex 23+i.3
```

Character arrays have a trailing null that is padded with nulls to a 4 byte boundary. This format of J array data will not change over releases with the possible exception of the nulls after character data. Do not depend on the character trailing nulls!

## **DDE**

[DDE Overview p192](#)

[Server and Client p193](#)

[DDE Conversations p194](#)

[J Commands & Events p195](#)

[Communication Protocol p196](#)

[Examples p197](#)



## DDE Overview

*Dynamic Data Exchange* allows Windows programs to communicate with each other.

DDE allows two Windows programs to exchange data by posting messages to each other using a standard protocol. J provides a comprehensive set of DDE commands so that you can communicate between J and any other Windows program supporting DDE, including another copy of J. You may have up to 20 DDE conversations at a time.

For example, with DDE you could set up J as a calculation server, so that another Windows program could send it a sentence for evaluation and receive back the result.

The DDE interface uses the Window Driver. Messages are *sent* with Window Driver commands, for example:

```
wd 'ddereq jserver calc res'
```

Messages are *received* as a Windows event, and require handlers, named `sys_eventname`, for each event.

For example, here is a typical result of `wd 'q'`, following a `ddepoke` event:

```
wdq
+-----+-----+
|syshandler|sys_handler|
+-----+-----+
|sysevent  |sys_ddepoke|
+-----+-----+
|sysdefault|sys_default|
+-----+-----+
|systopic  |top         |
+-----+-----+
|sysitem   |item        |
+-----+-----+
|sysdata   |+/\i.10     |
+-----+-----+
```

To respond to a ddepoke, you define a handler named `sys_ddepoke`, for example the following is used in file `system\packages\dde\server2.ijs`:

```
sys_ddepoke=: 3 : 0
sysdata=: sysdata -. CRLF,TAB
write0 sysdata
if. sysdata -: 'exit' do. return. end.
if. sysdata -: 'off' do.
    delay 1
    signoff ''
end.
try. val=: ".sysdata
catch. val=: 'unable to execute: ',sysdata end.
if. FORMAT do. val=: ":val end.
writel val
topitem=: topitem,systopic;sysitem;<val
)
```

Most Windows applications support DDE in some form or another. However, not all applications support the complete set of DDE commands, and in some cases the command usage is not standard. With J, you write programs to handle the DDE interface, and therefore you can tailor the J side of the protocol to fit the other application's requirements.

In this chapter we discuss general principles of DDE with examples of a J to J DDE link, and then discuss a link between J and *Microsoft Word for Windows*. Several other examples, including a link to *Microsoft Visual Basic* will be found in the directory: `system\examples\dde`.

## Server and Client

The two parties to a DDE conversation (*connection* or *link*) are known as the *server* and *client*. The server makes itself available by registering itself with Windows. The client initiates the conversation by sending a message to the server.

The DDE protocol is asymmetric: the client initiates the conversation, sends messages and instructions to the server, and terminates the conversation. The server cannot by itself initiate a conversation, or send messages except in response to a client request.

## DDE Conversations

There are three types of DDE conversations, known as *hot*, *warm* and *cold*:

- in a hot link, the client asks that whenever some data changes, the server send it the new data.
- in a warm link, the client asks that whenever some data changes, the server send it an indication that the data has changed (but not the new data itself).
- in a cold link, the server never notifies the client whenever data has changed.

Programs to handle each of these will be slightly different, and you will need to know what type of conversation the other application supports.

Hot and warm links are also known as *advise loops*.

A warm link is essentially a combination of a hot and cold link - the hot link advises of a change in the data item, the cold link is used to send the data item.

A conversation uses three identifiers: *service*, *topic* and *item*. A conversation may also move *data*:

- **service** identifies the server application, and is typically set by the server
- **topic** is a high level identifier of the information being exchanged
- **item** is a low level identifier of the information being exchanged
- **data** is the data being exchanged (as character data)

Service, topic, and item names are case-insensitive.

Some applications require specific topic and item names be used; others do not care. A commonly used topic is the *system* topic, which may also have a *sysitems* item, both are used to provide information about a server application to a client.

For example, Lotus 123 for Windows has a system topic:

```
wd 'ddereq 123w system sysitems'
SysItems Topics Formats RangeNames Selection Status
```

```
wd 'ddereq 123w system status'  
Ready
```

Some applications such as spreadsheets allow data to be exchanged in clipboard format. The utility `clipfmt` in script **format.ijs** will convert a list or table into clipboard format.

With spreadsheets, it is typical for the topic to be the worksheet name, and the item to be the cell or range name. For example, suppose `data` is a table of shape two by three, then this could be written to an Excel worksheet `sheet1` as follows:

```
txt=. clipfmt data  
wd 'ddepoke excel sheet1 r1c1:r2c3 *',txt
```

Some servers support a DDE *execute* command, allowing the client to use the server's command language directly. However, servers that support *execute* usually do so only in the system topic, and then only in a limited fashion, for example to run menu commands such as File/Open. For non-trivial tasks it is usually best to create a macro in the application and then use the *execute* command to run the macro. Most applications follow the conventions for *execute* expressions indicated in the following:

```
[open("sample.xlm")]  
[run("r1c1")]  
[connect][download(query1,results.txt)][disconnect]
```

## J Commands and Events

The following summarizes the Window Driver commands and event types. Here:

S

is the service name

T

is a topic

I

is an item

D

is some data *Server commands:*

ddename S

set service name S

ddereqd D

provide data D to a ddereq event. It must be the first command given after a ddereq event is received.

ddeadvise T I D

provides data  $D$  to advise loops with given topic and item. It returns 0 if the loop is no longer active.*Server events:*

ddepoke

data has been sent

ddeex

execute command string

ddereq

request for data

ddestart

request to start advise loop*Client commands:*

ddecons

return conversation names (service and topic)

ddedis [S [T]]

disconnect

ddeex S T D

execute command in server

ddepoke S T I D

poke data to server

ddereq S T I

request data from server

ddestart S T I

start advise loop

ddestop S T I

top advise loop

*Client events:*

result

answer to ddereq query

ddeadvise

new data from advise loop



## Communications Protocol

### *Opening the conversation:*

The client always initiates the conversation by sending a message to the server. The server must be ready to respond. How the server does this depends on the application.

For J to act as a server, you must have first loaded J and set the service name, either from the command line or using the `ddename` command. You also need event handlers for the events you want to respond to.

### *Client commands:*

The client can send four types of messages:

- a *ddepoke* message sends data to the server. The server does not respond. How the server treats this data depends on the application. If the server is a spreadsheet, the data may be written to the current worksheet; if the server is J, it may be treated as a sentence to be executed.
- a *ddeex* message sends a command to the server. The server does not respond. How the server treats this depends on the application, but it would typically be used to invoke a macro command in the server. A J server does not have to recognise this message, since a *ddepoke* message can be used instead.
- a *ddereq* message sends a request for a data item to the server. The server responds with the data item.
- a *ddestart* message sets up an advise loop, asking the server to advise the client whenever a specific data item has changed. The *ddestop* message terminates the advise loop. Once an advise loop has been initiated, the client must be prepared to accept messages from the server at any time.

### *Closing the conversation:*

A conversation is closed whenever either application terminates. A client can also close the conversation by issuing a *ddedis* command (received by Windows, but not sent to the server).

## Examples

The following examples illustrate. First we set up a J to J connection manually:

Load two copies of J, and arrange the windows so that both are visible at the same time. You may find it helpful to minimize Program Manager to reduce screen clutter.

In one window (the server), define the service name as *jserver*:

```
wd 'ddename jserver'
```

In the other window (the client), use the ddepoke command to send data to the server:

```
wd 'ddepoke jserver abc xyz mydata'
```

In the server, check the value of wdq

```
wdq
+-----+-----+
|syshandler|sys_handler|
+-----+-----+
|sysevent  |sys_ddepoke|
+-----+-----+
|sysdefault|sys_default|
+-----+-----+
|systopic  |abc        |
+-----+-----+
|sysitem   |xyz        |
+-----+-----+
|sysdata   |mydata     |
+-----+-----+
```

This indicates the character string *mydata* was sent to the J server, with topic *abc* and item *xyz*.

Now terminate the connection from the client:

```
wd 'ddedis'
```

In practice, you will want to set up a protocol that will enable client and server to communicate back and forth. To illustrate this we describe typical hot and cold links that use J as an execution server.

### J to J Hot Link

The script **system\packages\dde\server1.ijs** implements the server side of a DDE hot link. The protocol is as follows:

- the server defines its service name to be: *jserver*
- the client issues a `ddestart` command, using any topic and item
- the client issues a `ddepoke` for this topic and item, with the data being an executable J sentence
- the server executes the sentence, and returns the result with a `ddeadvice` command
- the client can issue a `ddepoke` repeatedly. If the data is `close` the topic and item are closed. If the data is `exit` the J server program is exited. If the data is `off` the J server task is terminated.

If you have not already done so, load 2 copies of J and minimize Program Manager to reduce screen clutter. In one copy of J (the server), enter:

```
load 'system\packages\dde\server1.ijs'
```

If you wish, you could load one copy of J, and enter the following command to load the second copy as a server (change the directory reference as appropriate):

```
wd 'winexec "\j3\j.exe system\packages\dde\server1.ijs"'
```

You could also create a new Shortcut or Program Manager item, which loads J with this script file as its profile file.

The server displays a Windows dialog box, which will be used to illustrate the conversation. If you wish, you can also minimize the server window.

In the other copy of J (the client), enter the following to initialize the client

```
load 'system\examples\dde\client1.ijs'
```

You should see some J code executed in the server window, and the result in the client session.

You can now send J sentences to be evaluated:

```
cmd 'i.4 5'
0 1 2 3 4
5 6 7 8 9
10 11 12 13 14
15 16 17 18 19
```

cmd sends its argument using ddepoke:

```
cmd=: 3 : 0
wd 'ddepoke jserver top item *',y.
)
```

The server window shows the conversation.

Try other expressions, for example, to create and use names:

```
cmd 'mydata=: 2 3 5 7'
2 3 5 7
```

```
cmd '#mydata"'
4
```

To close the server, enter:

```
cmd 'off'
```

(The server does not respond.)

J to J Cold Link

The script **system\packages\dde\server2.ijs** implements the server side of a DDE cold link. The protocol is as follows:

- the server defines its service name to be: *jserver*
- the client issues a ddepoke for any topic and item, with the data being an executable J sentence
- the server executes the sentence, and saves the result with the topic and item
- the client issues a ddereq command for the topic and item
- the server responds with the corresponding result
- the client can issue a ddepoke and ddereq repeatedly. If the ddepoke data is `exit` the J server program is exited. If the data is `off` the J server task is terminated.

As before, load 2 copies of J and minimize Program Manager. In the server copy of J, enter:

```
load 'system\packages\dde\server2.ijs'
```

then minimize the server window.

To initialize the client copy of J:

```
load 'system\examples\dde\client1.ijs'
```

Use `cmd` to set expressions:

```
cmd 'i.4 5'  
0 1 2 3 4  
5 6 7 8 9  
10 11 12 13 14  
15 16 17 18 19
```

`cmd` sends its argument using `ddepoke`, and retrieves the result using `ddereq`:

```
cmd=: 3 : 0  
wd 'ddepoke jserver top item *',y.  
wd 'ddereq jserver top item'  
)
```

Try other expressions, then to close the server, send:

```
cmd 'off'
```

(The server does not respond.)

Note that the hot link protocol is simpler than the cold link protocol, since once it is set up, the client need only issue one command to send a sentence and retrieve the result. Also, the ddereq command used with the cold link may time out if the server is not ready to send the data item requested.

[Calling DLLs p199](#)

[cd Domain Error & GetLastError p200](#)

[Memory Management p201](#)

[Calling J.DLL p202](#)

## Calling Procedures in Dynamic-link Libraries (DLLs)

**Calling procedures incorrectly can crash your system or corrupt memory.**

To learn how to call DLLs, run Labs 'DLL: Writing and Using a DLL' and 'DLL: Using System DLLs (file examples)'.

A DLL is a file (usually with extension .dll) that contains procedures. J can call DLL procedures.

Win32 API system services are provided by system DLLs such as *kernel32*. You can also use 3rd party DLLs or DLLs you write yourself.

Script `main\dll.ijs ( load 'dll' )` defines utilities for working with DLLs.

Verb `cd` calls a procedure. The form is:

```
'filename procedure [+][%] declaration' cd parameters.
```

- **filename** is the name of the DLL. If a suffix is not provided, .dll is used. The search path for finding a filename that is not fully qualified involves many directories and is different on each platform. Except for system DLLs, a fully qualified filename is recommended.
- **procedure** is the case-sensitive name of the procedure to call. A procedure name that is a number is specified by # followed by digits. Win32 API procedures that take string parameters are documented under a name, but are implemented under the name with an A suffix for 8 bit characters and a W suffix for 16 bit characters. For example, `CreateFile` is documented, but the procedures you call are `CreateFileA` or `CreateFileW`. A procedure returns a scalar result and takes 0 or more parameters. Parameters are passed by value or by a pointer to values. Pointer parameters can be read and set.
- **+** option selects the alternate calling convention. The calling convention is the rules for how the result and parameters are passed between the caller and the procedure. Using the wrong one can crash or corrupt memory. J supports two: `__stdcall` and `__cdecl`. `__stdcall` is used by the Win32 API and most procedures. `__cdecl` is the standard C calling convention and is used for some procedures.



\_\_stdcall is the standard cd calling convention and \_\_cdecl is the alternate.

- **%** option does an fpreset (floating-point state reset) after the call. Some procedures leave floating-point in an invalid state that causes a crash at some later time. DLL's built with Delphi likely have this problem. If J crashes on simple expressions after calling a procedure, try adding the % option.
- **declaration** is a set of blank delimited codes describing result and parameter types:

c

character (1 byte)

s

short integer (2 byte)

i

integer (4 byte)

f

short floating-point (4 byte)

d

floating point (8 byte)

j

complex (16 byte) (not as result)

\*

pointer

n

no result (result, if any, is ignored and 0 is returned)

The first declaration type describes the result and the remaining ones describe the parameters in the `cd` right argument.

The `c i d` and `j` types are native J types and the `s` and `f` types are not. Scalar `s` and `f` values are handled as `i` and `d` types. Pointer `s` and `f` parameters are handled as character data.

The `*` type is a pointer to values. A `*` can be followed by `c s i f d` or `j` to indicate the type of values. The DLL can read from this memory or write to it.

A scalar type (`c s i f d j`) must have a scalar parameter. A pointer type (`* *c *s *i *f *d *j`) must have either a non-scalar parameter of the right type, or a boxed scalar integer that is a memory address.

J boolean data is stored as 1 byte values. Boolean parameters are automatically converted to integers.

The `mema` result ([Memory Management p201](#)) can be used as a `*` type parameter. A memory address parameter is a boxed scalar. The NULL pointer is `<0`.

The `cd` right argument is a list of enclosed parameters. An empty vector is treated as 0 parameters and a scalar is treated as a single parameter.

The `cd` result is the procedure result catenated with its possibly modified right argument.

For example, the Win32 API procedure `GetProfileString` in `kernel32` gets the value of the `windows/device` keyword.

```
a=: 'kernel32 GetProfileStringA s *c *c *c *c s'
b=: 'windows';'device'; 'default'; (32$'z');32
a cd b
+---+-----+-----+-----+-----+-----+-----+-----+
|31|windows|device|default|HP LaserJet 4P/4MP,HP PCL5MS,LPT |32|
+---+-----+-----+-----+-----+-----+-----+-----+
```

The first type `s` indicates that the procedure returns a short integer. The first pointer names a section. The second pointer names a keyword. The third pointer is the

default if the keyword is not found. The fourth parameter is where the keyword text is put. The fifth parameter is the length of the fourth parameter.

If the GetProfileStringA declaration was wrong, say a d result instead of s, it would crash your system. If the fifth parameter was 64 and the keyword was longer than the 32 characters allocated by the fourth parameter, the extra data would overwrite memory.

Procedures are usually documented with a C prototype or a Visual Basic declaration. The C prototype and VB declaration for GetProfileString are:

```
DWORD GetProfileString(  
LPCTSTR lpAppName,      // address of section name  
LPCTSTR lpKeyName,      // address of key name  
LPCTSTR lpDefault,      // address of default string  
LPTSTR lpReturnedString,// address of destination buffer  
DWORD nSize             // size of destination buffer  
);
```

```
Declare Function GetProfileString Lib "kernel32"  
Alias GetProfileStringA"  
ByVal lpAppName As String,  
ByVal lpKeyName As String,  
ByVal lpDefault As String,  
ByVal lpReturnedString As String,  
ByVal nSize As Long)  
As Long
```

J declaration types and some corresponding C and VB types are:

J

C/Visual Basic

c

char, byte, bool

s

short int, word, %

i

int, long int, dword, &

f

float, !

d

double, #

\*

char\*, int\*, LP..., void\*, \$

n

void

`cd f ' '`  unloads all DLLs that `cd` has loaded. A loaded DLL is in use and attempts to modify it will fail. If you are developing and testing a DLL you must run `cd f ' '`  before you can build and save a new version.

Release J4.02 introduced some incompatible changes:

1. scalar error results are now domain errors (`cd er ' '`  provides details)
2. `*m` no longer supported
3. `fpreset` no longer always done (explicit `%` flag)
4. `*`  memory address must be boxed scalar, not a scalar

## **cd Domain Error and GetLastError Information**

`cderr` ' ' returns information about a cd domain error:

0 0 - no error

1 0 - file not found

2 0 - procedure not found

3 0 - too many DLLs loaded (max 20)

4 0 - parameter count doesn't match declarations

5 x - declaration x invalid

6 x - parameter x type doesn't match declaration

`cderrx` ' ' returns GetLastError and text from the last cd.

## Memory Management

Some DLL procedures return pointers to memory or require parameters of pointers to memory that cannot be provided by referencing a J array. The following verbs, defined in `main\dll.ijs`, are provided to allocate, free, read and write memory:

```
mema      allocate bytes of memory

memr      read bytes from memory (as type)

memw      write bytes to memory (from type)

memf      free memory allocated by mema
```

`mema` allocates memory. The result is an integer memory address. A 0 result indicates the allocation failed. For example:

```
address=: mema length
```

`memf` frees memory. The argument could be a `mema` result or pointer returned by a procedure. A 0 result is success and a 1 is failure.

```
memf address
```

`memr` reads data from memory. A `_1` count reads up to the first 0 (read a C string).

```
data=: memr address,byte_offset,count [,type]
```

`memw` writes data to memory. If type is char, count can be 1 greater than the length of the string left argument, in which case a 0 is appended (writing a C string).

```
data memw address,byte_offset,count [,type]
```

`memr` and `memw` type parameter is 2 4 8 or 16 for char integer float or complex. The default is 2. The count parameter is a count of elements of the type.

## Calling J.DLL

The J DLL can be called by any program that can call DLLs.

Since J.DLL is itself used by the J session, you need to make a copy of J.DLL first before calling it from J; for example, copy it to file JJ.DLL.

File system\examples\data\jdll.h. gives C prototypes for J procedures.

Script system\examples\dll\jdll.ijs. gives examples of calling the J DLL from J.

Use procedure JDo to execute a sentence. For example, the following writes text abc to file t1.txt:

```
load 'dll files'
cmd=: ''abc'' 1!:2 <'t1.txt'' NB. example sentence
'jj.dll JDo i *c' cd <cmd NB. send to J DLL
++-----+
|0|'abc' 1!:2 <'t1.txt'|
++-----+
fread 't1.txt' NB. check file was
written
abc
```

Use procedure JGetM to retrieve a J variable. The cd right argument is a name, followed by 4 pointers, which will correspond to the result datatype, rank, pointer to shape, and pointer to values. For example:

```
'jj.dll JDo i *c' cd <'ABC=: i.5' NB. define ABC
++-----+
|0|ABC=: i.5|
++-----+

'jj.dll JGetM i *c *i *i *i *i' cd 'ABC';4#<,0
++-----+
|0|ABC|4|1|13496196|13496500|
++-----+
```

The 6 items in the result are: error code (0=success), name, datatype (4=integer), rank(1), pointer to shape, and pointer to values.

The pointers refer to memory addresses within the J DLL. You should reference their values before calling the J DLL again, as further calls may invalidate the memory addresses. Use function `memr` to read memory and `ic` to convert to J integers. For example, the shape is:

```
_2 ic memr 13496196 0 4
5
```

Once the result datatype and shape are known, you can read the values, again using `memr`, and convert to a J variable.

File `system\examples\dll\jdll.ijs` defines functions that perform the necessary conversions. For example:

```
load 'system\examples\dll\jdll.ijs'
jdo 'ABC=: i.3 4'
++-----+
|0|ABC=: i.3 4|
++-----+

jget 'ABC'
0 1 2 3
4 5 6 7
8 9 10 11

jcmd 'q: 123456'
2 2 2 2 2 2 3 643
```



## **ODBC**

[Overview p204](#)

[The SQL Language p205](#)

[Installing ODBC p206](#)

[Connection & Statement Handles p207](#)

[Data Driver p208](#)

[Listing the Data Sources p209](#)

[ODBC error messages p210](#)

[Data Source Connection p211](#)

[Selecting & reading data p212](#)

[Updating a record p213](#)

[Creating a new file p214](#)

[SQL Statements p215](#)

[SQL Elements p216](#)

[SQL Reserved Words p217](#)

## ODBC Data Driver Overview

The Open Database Connectivity (ODBC) interface allows applications to access data from database management systems (DBMS), using Structured Query Language (SQL) expressions.

J Win9x/NT only supports 32-bit ODBC drivers.

The J/ODBC interface uses the *Data Driver* verbs, defined in script dd.ijs.

The processing required by the application is essentially independent of the DBMS. For example, if you have written programs to access dBase files using ODBC, then you can use similar programs to access Paradox, or indeed any other DBMS.

The application, and the driver programs that access the DBMS, are physically separate. To access a DBMS, you need only ensure a driver for that DBMS is available. There are ODBC drivers for virtually all commercial DBMS, and J itself is distributed with drivers for several popular systems, including Access, Btrieve, dBase, Excel, FoxPro, Oracle, Paradox and SQL Server.

Note that the DBMS need not itself support SQL, for example dBase does not. All that matters is that the ODBC DBMS *driver* is available.

An ODBC DBMS driver, together with information on where its datasets are stored, is typically referred to as a *data source*.

Between the application and the data there are 3 layers, though they appear to the application as a single unit:

**J application**

**Data Driver**

**ODBC Manager**

**Data Source**

**Data Source**

**Data Source**

**Data**

**Data**

**Data**

The application sends requests to the Data Driver (using the verbs defined in script dd.ijs), which converts them into a standard format and sends them to the ODBC Manager. The ODBC Manager then sends them to the appropriate Data Source, and is also responsible for ensuring the required drivers are loaded. The Data Source drivers then handle the data access.

## **The SQL language**

SQL, or Structured Query Language, is a widely accepted protocol used for data access. It is an ANSI standard with SQL-92 being the most recent specification. A summary of the language appears in Appendix B.

The language is defined with various levels, listed in the appendix as minimum, core and extended. All SQL servers support at least the minimum level, some may also support core or extended levels, or extensions of their own. ODBC DBMS drivers are distributed with Help files that list the functionality of the driver, plus other useful information - when you install the drivers you should also print out the Help files for reference.

## Installing ODBC

ODBC must be installed on your system. An easy way to check for this is to look for the 32bit ODBC icon in Control Panel. ODBC is installed automatically with many Microsoft applications.

If you do not have ODBC installed, you may be able to obtain ODBC drivers from the Microsoft web site [www.microsoft.com](http://www.microsoft.com). Search for the ODBC FAQ (at the time of writing this is at [www.microsoft.com/data/odbc/faq\\_odbc.htm](http://www.microsoft.com/data/odbc/faq_odbc.htm)).

To follow the examples in this chapter, you should install the dBase driver, even if you do not intend to use it later. When you install this driver, or any other driver, you will be prompted to enter the *data source name*. The data source indicates the DBMS driver and where the data files are stored. Note that it is *not* the name of a specific data file as may be imagined.

The examples here use the dBase driver, and files stored in J subdirectory examples\data, and the data source name used is *jdata*. To set this up, in Program Manager, select the ODBC icon in Control Panel. Use Add or Setup to create a panel as shown:

Note that you should at least have one data source name as shown above, but you can set up multiple data source names, using the same driver but typically with a different directory.

Although the data source name specifies a directory, this is in fact used by the dBase driver as the default directory. You can override this default either by specifying another directory when you connect to the dBase driver, or afterwards when you specify a file with its full pathname. However, note this capability is provided by extensions to ODBC in the dBase driver, which are not necessarily found in other ODBC drivers.

## Connection and Statement Handles

You start accessing a database by first opening a connection to the data source. The result is a number, called the *connection handle*. Subsequent requests to the data source use this connection handle.

When you send a request to select some data, the result is also a number, called the *statement handle*. Again, subsequent requests that reference this selection use the statement handle. There may be more than one statement handle associated with a connection handle. You may also have a statement handle not associated with a connection handle, where the request made was not specific to a particular data file.

In both cases, you should close the handle to free up resources when you are finished with it. For example, to read records from a file, you typically:

- open a connection to the data source, returning a connection handle
- make a selection on a specific file using the connection handle, returning a statement handle
- read the records, using the statement handle
- close the statement handle
- close the connection handle

You must always close a connection handle explicitly. However, a statement handle may be closed if you have fetched all records selected - see the discussion of `ddfet` below. In any case, it is good practice to always close the statement handle explicitly, as there is no harm done if you try to close a handle that had already been closed.

## Data Driver

ODBC access is provided by the Data Driver verbs, that are defined in script `dd.ijs`.  
First load this file:

```
load 'dd'
```

The Data Driver verbs may be summarized as follows:

Here, *ch* refers to a connection handle, and *sh* a statement handle. Note that SQL commands are not case-sensitive.

**ddcnm** `r=. ddcnm sh`

Column names of selected data

**ddcnt** `r=. ddcnt ''` (available in J Win95 only)

Rowcount of last `ddsql` command

**ddcol** `r=. 'tdata' ddcol ch`

Column names and attributes in a database

**ddcom** `r=. ddcom ch`

Commit a transaction (after a `ddtrn`)

**ddcon** `ch=. ddcon 'dsn=jdata'`

Connect to ODBC data source name. The result is a connection handle. The argument can set several parameters, separated by semicolons. Some are supported by all databases, and others have a meaning only for specific databases. Parameters recognized by most database systems are:

**dsn**

ODBC data source name

**dlg**

dlg=1 prompts for a connection string with a dialog box with entries for User Id and Password (not supported by the distributed dBase driver)

### **uid**

user name

### **pwd**

user password

### **modifysql**

set to 1 (the default) to use ODBC SQL grammar. Set to 0 to use native database grammar.

### **rereadafterupdate**

set to 1 to force a re-read of a record after an update. This is useful for retrieving auto-updated values such as timestamps.

### **rereadafterinsert**

set to 1 to force a re-read of a record after an insert

For example:

```
ch=. ddcon'dsn=mydata;uid=george;pwd=sesame'
```

```
dddis r=. dddis ch
```

Closes connection handle (disconnects from the data source)

```
ddend r=. ddend sh
```

Closes statement handle

```
dderr r=. dderr ''
```

Return error message on last command. An error message is given when a data driver verb returns \_1.

```
ddfet r=. ddfet sh,n
```

Fetch next records from selected data. Note that after you have read a record, the next fetch will not read it again. If you need to read it again, you must select it



again. For example:

```
r=. ddfet sh
```

fetch next record (same as `ddfet sh,1`)

```
r=. ddfet sh,5
```

fetch next 5 records

```
r=. ddfet sh,_1
```

fetch all remaining records.

If you fetch all remaining records using `ddfet sh,_1`, or if your fetch returns fewer records than you requested (i.e. the fetch reads past the end of the file), then `ddfet` closes the statement handle. Otherwise, the statement handle remains open, and you should explicitly close it if you have finished reading the file.

**ddfch** `r=. ddfch sh,n` (available in J Win95 only)

As `ddfet`, but returns data in columns

**ddrbk** `r=. ddrbk ch`

Discards (rollbacks) a transaction (after a `ddtrn`)

**ddsel** `sh=. 'select * from tdata' ddsel ch`

Select data from a database, returning a statement handle

**ddsql** `r=. 'create table mydata' ddsql ch`

Execute an SQL statement

**ddsrc** `r=. ddsrc ''`

In J Win95:

Data source names available from the ODBC manager.

These names can be used as the `dsn=` argument to `ddcon`.

In J Win31:

a statement handle that can be used with `ddfet` to return data source names.

**ddtbl** sh=. ddtbl ch

Returns a statement handle for tables in the data source. Some ODBC drivers, including the distributed dBase driver, do not support this service and the result will be empty.

**ddtrn** r=. ddtrn ch

Begin a transaction on a connection. Subsequent actions are not committed to the database until a `ddcom` is done. Actions since the `ddtrn` can be discarded by doing a `ddrbk` (rollback).

## Listing the Data Sources

The list of data sources that are specified in the ODBC Control Panel dialog box can be retrieved under program control. The verb `ddsrc` requests this information. In J Win95, the result is the list of data sources; in J Win31, the result is a statement handle that may be used with `ddfet` to return the list of data sources.

```
ddsrc''
```

MS Access 7.0 Database	Microsoft Access Driver (*.mdb)	
Excel Files	Microsoft Excel Driver (*.xls)	
FoxPro Files	Microsoft FoxPro Driver (*.dbf)	
Text Files	Microsoft Text Driver (*.txt; *.csv)	
dBASE Files	Microsoft dBase Driver (*.dbf)	
jdata	Microsoft dBase Driver (*.dbf)	

The result will depend on the drivers you have set up. Note that you should see an entry for *jdata*, which is the data source name you assigned to the examples\data subdirectory.

The result has 2 columns: the data source name, and a description of the driver.

## ODBC error messages

The data driver verbs that open and close connections, and send SQL statements, all return a number. Typically, if the number is positive, it is a handle. If the number is 0, it means the operation completed successfully (but the function returns no handle). If the number is `_1`, it means there was some error. You can get more information about the error using `dderr`. For example, try closing a non-existent statement handle:

```
ddend 42
_1
dderr''
ISI04 Bad statement handle
```

## Connecting to a data source

The `ddcon` command connects to a data source returning a connection handle, using the form:

```
[ch= . ddcon 'dsn=jdata'
1
```

We have now connected to a data source, but not yet to a data file. The file we will use is *tdata*, and the next statement uses `ddcol` to retrieve the column names and attributes for this file:

```
$cols= . 'tdata' ddcol ch
7 13
```

```
3 4 7 {"1 cols
+-----+-----+-----+
| Column|Type|Type_Name|
+-----+-----+-----+
| NAME  |1   |CHAR      |
+-----+-----+-----+
| SEX   |1   |CHAR      |
+-----+-----+-----+
| DEPT  |1   |CHAR      |
+-----+-----+-----+
...
```

## Selecting and reading data

To read data from a file, you first select the data you want to read using the `ddsel` verb, which returns a statement handle. You then use `ddfet` or `ddfch` to fetch the records.

The left argument of `ddsel` is a SQL selection expression. Here are typical examples:

Select all records (\* means all columns):

```
sh=. 'select * from tdata' ddsel ch
```

Fetch the first 3 records:

```
ddfet sh,3
```

```
+-----+-----+-----+-----+-----+
|MACDONALD B   |F|D101|1.95906e7|1.97805e7|32591|
+-----+-----+-----+-----+-----+
|GENEREUX S    |F|D103|1.94503e7|1.96602e7|95415|
+-----+-----+-----+-----+-----+
|KOEDEL R      |M|D101|1.93711e7|1.98009e7|63374|
+-----+-----+-----+-----+-----+
```

Fetch the next record:

```
ddfet sh
```

```
+-----+-----+-----+-----+-----+
|KELLER J      |F|D101|1.95105e7|1.97404e7|48898|
+-----+-----+-----+-----+-----+
```

Close the statement handle:

```
ddend sh
```

You should always close the statement handle when you no longer need it.

However to avoid repetition, the remaining examples do not show this.

Select males with salary exceeding 40000:

```
sel=.'select * from tdata where sex='M' and salary >= 40000'
sh=. sel ddsel ch
ddfet sh,4
```

```
+-----+-----+-----+-----+
|KOEDEL R      |M|D101|1.93711e7|1.98009e7|63374 |
+-----+-----+-----+-----+
|NEWTON R      |M|D108|1.95601e7|1.97902e7|73368 |
+-----+-----+-----+-----+
|DINGEE S      |M|D103|1.9641e7 |1.98309e7|46877 |
+-----+-----+-----+-----+
|ROGERSON G    |M|D101|1.95712e7|1.98302e7|108777|
+-----+-----+-----+-----+
```

Select only the name, department and salary fields, where date of birth is before 1950:

```
sel=.'select name,dept,salary from tdata where dob<19500000'
```

Fetch the first such record:

```
ddfet (sel ddsel 1),1
```

```
+-----+-----+
|GENEREUX S    |D103|95415|
+-----+-----+
```

Use ddfch to return data in columns:

```
[a=. ddfch 1005,_1
```

```
+-----+-----+-----+-----+
|MACDONALD B   |F|D101|1.95906e7|1.97805e7|32591|
|GORDON E      |F|D103|1.95202e7|1.97908e7|29960|
|BAUERLEIN J   |F|D103|1.96204e7|1.98409e7|33668|
|CHESHER D     |F|D103| 1.9561e7|1.98408e7|35184|
+-----+-----+-----+-----+
```

```
(;:'name sex dept dob doh salary')=. a
salary
32591
```

29960  
33668  
35184



## Updating a record

To update a record, you process an SQL update statement. Here we update the salary field for ABBOTT K.

First read the record to see the current value (the salary is 50817):

```
sel=. 'select * from tdata where name=''ABBOTT K'''
ddfet sel ddsel ch
```

ABBOTT K	M	D103	1.9631e7	1.98309e7	50817
----------	---	------	----------	-----------	-------

Next we process an update statement:

```
us=. 'update tdata set salary=45000 where name=''ABBOTT K'''
us ddsql ch
```

Finally, we read the record again, to ensure the update was successful:

```
sel=. 'select * from tdata where name=''ABBOTT K'''
ddfet sel ddsel ch
```

ABBOTT K	M	D101	1.9631e7	1.98309e7	45000
----------	---	------	----------	-----------	-------

## Creating a new file

To create a new file, use the SQL *create* command, with a list of the column names and attributes in the new data set. Here we create file *test*, with columns for name and salary.

First, drop test in case it already exists (the `_1` result means the table was not found):

```
'drop table test' ddsql ch
_1
```

Now create the new file:

```
'create table test (name char(12),sal numeric)' ddsql ch
0
```

Add a record:

```
t=. 'insert into test (name,sal) values (''Neumann,E'',40000)'
t ddsql ch
0
```

Add another record:

```
t=. 'insert into test (name,sal) values (''James, P'',42000)'
t ddsql ch
0
```

Now read the file:

```
ddfet _1,~ 'select * from test' ddsel ch
+-----+-----+
|Neumann, E |40000|
+-----+-----+
|James, P   |42000|
+-----+-----+
```

## SQL Statements

The following SQL statements define the base ODBC SQL grammar.

Statement	Min Core Ext
alter-table-statement ::= ALTER TABLE base-table-name { ADD column-identifier data-type   ADD (column-identifier data-type [, column-identifier data-type]... ) }	X
create-index-statement ::= CREATE [UNIQUE] INDEX index-name ON base-table-name ( column-identifier [ASC   DESC] [, column-identifier [ASC   DESC] ]... )	X
create-table-statement ::= CREATE TABLE base-table-name-1 (column-element [, column-element] ...) column-element ::= column-definition   table-constraint-definition column-definition ::= column-identifier data-type DEFAULT default-value [column-constraint-definition [,column-constraint-definition]...] column-constraint-definition ::= NOT NULL   UNIQUE   PRIMARY KEY )   REFERENCES base-table-name-2 referenced-columns   CHECK (search-condition) default-value ::= literal   NULL   USER table-constraint-definition ::= UNIQUE (column-identifier [, column-identifier] ...)   PRIMARY KEY (column-identifier [, column-identifier] ...)   CHECK (search-condition)	X

FOREIGN KEY referencing-columns REFERENCES base-table-name-2 referenced-columns		
create-view-statement ::=		X
CREATE VIEW viewed-table-name [( column-identifier [, column-identifier]... )] AS query-specification		
delete-statement-positioned ::=		X (v2)X
DELETE FROM table-name WHERE CURRENT OF cursor- name		X
delete-statement-searched ::=	X	
DELETE FROM table-name [WHERE search-condition]		
drop-index-statement ::=		X
DROP INDEX index-name		
drop-table-statement ::=	X	
DROP TABLE base-table-name [ CASCADE   RESTRICT ]		
drop-view-statement ::=		X
DROP VIEW viewed-table-name [ CASCADE   RESTRICT ]		
grant-statement ::=		X
GRANT { ALL   grant-privilege [, grant-privilege]... } ON table-name TO { PUBLIC   user-name [, user-name]... } grant-privilege ::=		
DELETE   INSERT   SELECT   UPDATE [( column-identifier [, column-identifier]... )]   REFERENCES [( column-identifier [, column-identifier]... )]		
insert-statement ::=	X	
INSERT INTO table-name [( column-identifier [, column- identifier]...)] VALUES (insert-value[,insert-value]... )		

insert-statement ::=					X
INSERT INTO table-name [( column-identifier [, column-identifier]... )]					
{ query-specification   VALUES (insert-value [, insert-value]... }					
ODBC-procedure-extension ::=					X
ODBC-std-esc-initiator [?]=call procedure ODBC-std-esc-terminator					
ODBC-ext-esc-initiator [?]=call procedure ODBC-ext-esc-terminator					
revoke-statement ::=					X
REVOKE {ALL   revoke-privilege [, revoke-privilege]... }					
ON table-name					
FROM {PUBLIC   user-name [, user-name]... }					
[ CASCADE   RESTRICT ]					
revoke-privilege ::=					
DELETE					
INSERT					
SELECT					
UPDATE   REFERENCES					
select-statement ::=					X
SELECT [ALL   DISTINCT] select-list					
FROM table-reference-list[					
[WHERE search-condition]					
[order-by-clause]					
select-statement ::=					X
SELECT [ALL   DISTINCT] select-list					
FROM table-reference-list					
[WHERE search-condition]					
[GROUP BY column-name [, column-name]... ]					
[HAVING search-condition]					
[UNION select-statement]...					
[order-by-clause]					
select-for-update-statement ::=					X X
SELECT [ALL   DISTINCT] select-list					
FROM table-reference-list					
[WHERE search-condition]					
FOR UPDATE OF [column-name [, column-name]...]					
				(v1)	(v2)

statement ::= create-table-statement	X
delete-statement-searched   drop-table-statement	
insert-statement   select-statement	
update-statement-searched	
statement ::= alter-table-statement	X
create-index-statement	
create-table-statement	
create-view-statement	
delete-statement-positioned	
delete-statement-searched	
drop-index-statement	
drop-table-statement	
drop-view-statement	
grant-statement	
insert-statement	
revoke-statement	
select-statement	
select-for-update-statement	
update-statement-positioned	
update-statement-searched	
statement ::= alter-table-statement	X
create-index-statement	
create-table-statement	
create-view-statement	
delete-statement-positioned	
delete-statement-searched	
drop-index-statement	
drop-table-statement	
drop-view-statement	
grant-statement	
insert-statement	
ODBC-procedure-extension	
revoke-statement	
select-statement	
select-for-update-statement	
statement-list	
update-statement-positioned	
update-statement-searched	

statement-list ::= statement   statement;statement-list			X
update-statement-positioned ::=		X	X
UPDATE table-name			
SET column-identifier = {expression   NULL}		(v1)	(v2)
[, column-identifier = {expression   NULL}]...			
WHERE CURRENT OF cursor-name			
update-statement-searched		X	
UPDATE table-name			
SET column-identifier = {expression   NULL }			
[, column-identifier = {expression   NULL}]...			
[WHERE search-condition]			

## Elements Used in SQL Statements

The following elements are used in the SQL statements listed previously .

### Element

**Min**

**Core**

**Ext**

approximate-numeric-literal ::= mantissaEexponent

mantissa ::= exact-numeric-literal

exponent ::= [+|-] unsigned-integer

X

approximate-numeric-type ::=

FLOAT

| DOUBLE PRECISION

| REAL

X

argument-list ::= expression | expression, argument-list

X



base-table-identifier ::= user-defined-name

X

base-table-name ::= base-table-identifier

X

base-table-name ::= [user-name.]base-table-identifier

X

between-predicate ::= expression [NOT] BETWEEN expression AND expression

X

binary-literal ::= {implementation defined}

X

binary-type ::= BINARY (length)  
| VARBINARY (length)  
| LONG VARBINARY.

X

character ::= {any character in the implementor's character set}

X

character-string-literal ::= '{character}...'

X

character-string-type ::=  
CHARACTER(length)  
| CHAR(length)

X

character-string-type ::=  
CHARACTER(length)  
| CHAR(length)  
| CHARACTER VARYING(length)  
| VARCHAR(length)

X

character-string-type ::=  
CHARACTER(length)  
| CHAR(length)  
| CHARACTER VARYING(length)  
| VARCHAR(length)

| LONG VARCHAR

X

column-identifier ::= user-defined-name

X

column-name ::= [table-name.]column-identifier

X

column-name ::= [{ table-name | correlation-name }.]column-identifier

X

comparison-operator ::= < | > | <= | >= | = | <>

X

comparison-predicate ::=  
expression comparison-operator expression

X

comparison-predicate ::= expression comparison-operator

{expression | (sub-query)}

X

correlation-name ::= user-defined-name

X

cursor-name ::= user-defined-name

X

data-type ::= character-string-type

X

data-type ::=  
character-string-type  
| exact-numeric-type  
| approximate-numeric-type

X

data-type ::=  
character-string-type  
| exact-numeric-type  
| approximate-numeric-type  
| binary-type

| date-type  
| time-type  
| timestamp-type

X

date-literal ::= 'date-value'

X

date-separator ::= -

X

date-type ::= DATE

X

date-value ::=  
years-value date-separator months-value date-separator days-value

X

days-value ::= digit digit·

X

digit ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

X

dynamic-parameter ::= ?

X

exact-numeric-literal ::=  
[+|-] { unsigned-integer [.unsigned-integer ]  
| unsigned-integer.  
| .unsigned-integer }

X

exact-numeric-type ::= DECIMAL(precision,scale)  
| NUMERIC(precision,scale) | SMALLINT | INTEGER

X

exact-numeric-type ::= DECIMAL(precision,scale)  
| NUMERIC(precision,scale) | BIT | SMALLINT | INTEGER  
| BIGINT

X

exists-predicate ::= EXISTS ( sub-query )

X

expression ::= term | expression {+|-} term

term ::= factor | term {\*/} factor

factor ::= [+|-]primary

primary ::= column-name

| dynamic-parameter

| literal

(continued)

| ( expression )

primary ::= column-name

| dynamic-parameter

| literal

| set-function-reference

| USER

| ( expression )

primary ::= column-name

| dynamic-parameter

| literal

| ODBC-scalar-function-extension

| set-function-reference

| USER

| ( expression )

X

X

X

hours-value ::= digit digit

X

index-identifier ::= user-defined-name

X

index-name ::= [index-qualifier.]index-identifier

X

Index-qualifier ::= user-defined-name

X

in-predicate ::=  
expression [NOT] IN {(value {, value}...) | (sub-query)}  
value ::= literal | USER | dynamic-parameter

X



## SQL Reserved Keywords

The following words are reserved for use in ODBC function calls. These words do not constrain the minimum SQL grammar; however, to ensure compatibility with drivers that support the core SQL grammar, applications should avoid using any of these keywords.

ABSOLUTE	DECLARE	INTEGER	REVOKE
ADA	DEFERRABLE	INTERSECT	RIGHT
ADD	DEFERRED	INTERVAL	ROLLBACK
ALL	DELETE	INTO	ROWS
ALLOCATE	DESC	IS	SCHEMA
ALTER	DESCRIBE	ISOLATION	SCROLL
AND	DESCRIPTOR	JOIN	SECOND
ANY	DIAGNOSTICS	KEY	SECTION
ARE	DICTIONARY	LANGUAGE	SELECT
AS	DISCONNECT	LAST	SEQUENCE
ASC	DISPLACEMENT	LEFT	SET
ASSERTION	DISTINCT	LEVEL	SIZE
AT	DOMAIN	LIKE	SMALLINT
AUTHORIZATION	DOUBLE	LOCAL	SOME
AVG	DROP	LOWER	SQL
BEGIN	ELSE	MATCH	SQLCA
BETWEEN	END	MAX	SQLCODE
BIT	END-EXEC	MIN	SQLERROR
BIT_LENGTH	ESCAPE	MINUTE	SQLSTATE
BY	EXCEPT	MODULE	SQLWARNING
CASCADE	EXCEPTION	MONTH	SUBSTRING
CASCADED	EXEC	MUMPS	SUM
CASE	EXECUTE	NAMES	SYSTEM
CAST	EXISTS	NATIONAL	TABLE
CATALOG	EXTERNAL	NCHAR	TEMPORARY
CHAR	EXTRACT	NEXT	THEN
CHAR_LENGTH	FALSE	NONE	TIME
CHARACTER	FETCH	NOT	TIMESTAMP
CHARACTER_LENGTH	FIRST		
NULL	TIMEZONE_HOUR		
CHECK	FLOAT		
NULLIF	TIMEZONE_MINUTE		
CLOSE	FOR	NUMERIC	TO
COALESCE	FOREIGN	OCTET_LENGTH	TRANSACTION
COBOL	FORTRAN	OF	TRANSLATE

COLLATE	FOUND	OFF	TRANSLATION
COLLATION	FROM	ON	TRUE
COLUMN	FULL	ONLY	UNION
COMMIT	GET	OPEN	UNIQUE
CONNECT	GLOBAL	OPTION	UNKNOWN
CONNECTION	GO	OR	UPDATE
CONSTRAINT	GOTO	ORDER	UPPER
CONSTRAINTS	GRANT	OUTER	USAGE
CONTINUE	GROUP	OUTPUT	USER
CONVERT	HAVING	OVERLAPS	USING
CORRESPONDING	HOURL	PARTIAL	VALUE
COUNT	IDENTITY	PASCAL	VALUES
CREATE	IGNORE	PLI	VARCHAR
CURRENT	IMMEDIATE	POSITION	VARYING
CURRENT_DATE	IN	PRECISION	VIEW
CURRENT_TIME	INCLUDE	PREPARE	WHEN
CURRENT_TIMESTAMP	INDEX	PRESERVE	WHENEVER
CURSOR	INDICATOR	PRIMARY	WHERE
DATE	INITIALLY	PRIOR	WITH
DAY	INNER	PRIVILEGES	WORK
DEALLOCATE	INPUT	PROCEDURE	YEAR
DEC	INSENSITIVE	PUBLIC	
DECIMAL	INSERT	RESTRICT	

## **OLE & OCX**

[Overview p219](#)

[J OLE Automation Server p220](#)

[J OLE/OCX Client p221](#)

[Examples p222](#)

[Tutorial: J OLE Server for Excel p223](#)

[Tutorial: J OLE Client to Excel p224](#)

## OLE & OCX Overview

OLE allows client programs to access server programs, enabling integration between a variety of software components. All OLE servers are usable by all OLE clients.

An OCX control, also known as an ActiveX control, or Microsoft Custom Control, is a special form of OLE control. The main difference is that OCX controls cause events, and are *in-process* server objects (DLL) whereas OLE controls are *local-server objects* (EXE).

Both OLE and OCX controls are application packages that can be accessed in a simple and consistent way from other Windows applications.

J Win95/NT supports OLE and OCX, enabling J to be used both as a calculation server and as a client for other Windows software such as Visual Basic and Excel. In particular, OLE makes it very easy to add the power of J to other applications.

### OLE

A key part of OLE is that a server can run in the same task, or as a separate task, or even as a separate task on another network connected machine. A server in the same task is an *in-process server*, a server in another task on the same machine is a *local server*, and a server on another machine is a *remote server*. An in-process server is the most efficient; the task can call the in-process server's routines almost as efficiently as its own routines.

Servers provide functions, called *methods*, that can be accessed by a client. For example, the J server provides a method `Do` which executes a J sentence; and a method `Get` which retrieves the value of a variable. If a client runs the method `Do 'x = . 3+5 '`, this causes the sentence to be executed in the J server. The client can then call method `Get` to retrieve the value of `x`.

*OLE Automation* is an interface supported by many clients and servers. OLE Automation gives a client an interface to a server that mimics the server's end-user interface. This allows a client to automate what would otherwise have to be done manually by a user. Although this is useful, the real power comes when the server has a general purpose programming environment that can be used to build complex

services that can be used from a client. OLE Automation then provides easy access from the client to the full power of the server.

J supports OLE Automation as follows:

***J OLE Automation server.*** There are two servers:

- *JDLLServer* is an in-process J server
- *JEXEServer* is a local J server

***J OLE Automation client.*** This supports servers such as Excel, that provide an OLE Automation interface to an application object that supports macro execution.

## J OLE Automation Server

There are two servers:

- JEXEServer is the full J development system and is intended for use in developing applications. This is the same J.EXE file that provides the regular J development system.
- JDLLServer is the J interpreter only, and is intended for runtime applications. This is in file J.DLL.

An EXE server runs as a separate application from the client, i.e. is a local server. JEXEServer provides the full J development system, which makes it easy to develop and debug applications. As a developer, you have full access to both the client and the J server environments.

A DLL is part of the client application and uses the same memory space, i.e. is an in-process server. A client accesses DLL services almost as efficiently as it accesses its own native services. The JDLLServer is not as convenient as JEXEServer for development purposes, but it is very efficient and is ideal for runtime applications.

Typically, you develop your application using JEXEServer, and run it using the JDLLServer.

Note that JEXEServer and JDLLServer are 32bit servers and are designed for 32bit clients. It may be possible to use JEXEServer from 16bit clients, but this is not officially supported.

J.DLL can be called directly as an ordinary DLL, without using OLE. For more information, see online help *DLLs and Memory Management*.

### Clients

Any application with OLE Automation controller support, such as Visual Basic, Delphi, Excel, or a Visual C++ application, can use JEXEServer and JDLLServer. Also, any application that can call 32bit DLLs can access JDLLServer.

## Registration

JEXEServer and JDLLServer must be registered with your system before they can be used. To do so, you run JREG.EXE , which is stored in the same directory as J.EXE, i.e. select Start/Run and enter:

c:\j401\jreg.exe (use the correct directory name)

If there are problems later when accessing the J servers it may be because they are no longer properly registered. You can always run JREG.EXE to register the servers again. In particular, you will need to do this if you move the J system files to another directory, or if for some reason, the Registry is damaged and you have to recover an old version.

## Using the J OLE Automation servers

The steps are fairly straightforward, but may differ in minor ways from one client to another.

You should be familiar with both the J system and the client before tackling them in a client/server combination.

First load the client application and ensure it references the J servers. You need only select the server you intend to use, but when experimenting, you should check both:

Once the J servers are referenced, you can check the methods available, which are as follows.

## Break

interrupt J execution

## Clear

erases all definitions in J

## Do

execute a J sentence

ErrorText/ErrorTextM

get error text (run after a J error)

Get/GetB/GetM

get the value of a J variable

IsBusy

returns 0 if J is ready to execute, else an error code

Log

display (1) or discard (0) the J EXE session log

Quit

causes J EXE server to close when last object is released

Set/SetB/SetM

set a value to a J variable

Show

show (1) or hide (0) the J EXE server

Transpose

return array data transposed

For details, see file system\examples\data\jdll.h.

Note that methods Show and Log are ignored by a JDLLServer.

Once the J servers have been referenced, their methods become available for use by the client. You should declare the J server as an object in the client, and can then reference that object in order to access the J server methods.



## J OLE/OCX Client

J interacts with OLE and OCX controls using the Window Driver. You need a parent window to display the control, and you then create and use a OLE/OCX control in much the same way as for any other child control. You can use both standard and OLE/OCX controls on the same form.

```
wd 'cc tree
ocx:comctl.treectrl.1'
wd 'cc xl oleautomation:excel.application'
```

OCX and OLE controls are not distributed with J. Some OCX and OLE controls are included with Windows, and with software such as Visual Basic. In some cases, these are subsets or earlier versions of the full systems obtainable from the original manufacturer. For serious use, we recommend you purchase the most recent versions of the controls, which will then include full documentation and support.

To see which OCX controls are available, create a new form using the Form Editor, then click New for new control, and select class OCX.

OLE controls are not supported by the Form Editor and should be added directly in the `form_run` verb. The names of the OLE controls that are available are not visible from J.

An OCX or OLE control is a child on a form that contains an object named *base*. This base object can contain other objects. For example the base object could contain a font object. An object is identified by the child id and the name of the object. The name *temp* refers to a contained object that has been returned by a method or property. The temp object is automatically released and reset whenever a new object is returned. The `oleid` command gives the temp object a name and makes it permanent.

Each control has a specific list of *methods*, *properties* and *events* (for OCX controls). The properties define the way the control is set up. The events are the events that the OCX control can generate. When you create a control, you typically set various properties, and register various events that you want to be notified of. When the control is in use, you retrieve information from the control by either

reading the value of its properties, or by receiving event notifications.

## **wd commands for OLE/OCX controls**

`oledlg id`

Run the OCX property dialog. The state can be saved with the `olesave` command.

`oleenable id eventname [bool]`

Enable/disable OCX event. You must enable an event in order to trigger an event in J.

`oleget id objectname property`

Return value of a property. Objectname is *base*, *temp*, or a name set with `oleid`. If the result is an object, it is set as the temp object. This allows a series of `wd` commands that use the temp object to get the next object.

`oleinfo id`

Return information about events, methods, properties, and constants.

`oleload id filename`

Initialize properties from a file created by `olesave`. An `oleload` should only be done once before it is shown.

`olemethod id objectname method parameters....`

Run a method.

, is an elided parameter. A `wd` parameter of , is the same as "", except it is treated as an elided parameter where appropriate.

Some methods distinguish between a numeric parameter and a string. A simple (not delimited) string that is an integer is treated as an integer. If you want 23 to be treated as a string, use "23";.

If the result is an object, it is set as the temp object.

An object parameter is indicated by a simple parameter of the form:

`object:formid.childid.objectname`

A picture object parameter is indicated by a simple parameter of the form:

`picture:filename`

`olesave id filename`

Save properties in a file that can be used to initialize a control after it is created.

```
oleset id objectname property value  
Set property value.
```

## Objectnames

An OLE or OCX base object can contain other objects.

For example, many OCX controls contain a Font object that is returned by the font property.

```
wd 'oleget ocx base font' get font object as temp
```

```
wd 'oleget ocx temp fontsize' get font size  
8.5
```

```
wd 'oleget ocx temp fontname' get font name  
Times New Roman
```

```
wd 'oleset ocx temp Times New Roman;oleset ocx temp fontsize 20'  
set font
```

## OCX Events

An OCX event is signalled as a button event. For example, form abc, OCX id spin, signals sysevent abc\_spin\_button. The name sysocx is assigned the OCX event name.

## Example: Spinbutton Control

To illustrate these commands, we use the Outrider Systems SPIN32.OCX, a simple control that is ideal for experimentation. We assume you have this control installed on your system.

First try reading the OCX information (if this fails, you do not have the SpinButton installed). This returns a text description of properties, methods and events:

```
load 'packages\ocx\ocxutil.js' load OCX utilities  
ocxinfo 'spin.spinbutton'  
NB. event: SpinDown  
NB. prototype: void SpinDown ()  
NB. help: Occurs when the user clicks one of the arrows...  
  
NB. event: SpinUp
```

...

Next, load the spinbutton demo, as follows: `load`  
`'examples\ocx\misc\spin.js'`

Click on the spinbutton to change the text field. Click on the About button to display the about box from Outrider, and the Dialog button to view and change the OCX properties.

Note that to see events from the spinbutton, you use the `oleenable` command when initializing the form, i.e.

```
spin_run=: 3 : 0
wd SPIN
showpay''
wd 'oleenable sx spindown'
wd 'oleenable sx spinup'
wd 'pshow;'
)
```

The About button invokes the OCX's `aboutbox` method:

```
spin_about_button=: 3 : 0
wd 'olemethod sx base aboutbox'
)
```

The Dialog button invokes the `oledlg` command:

```
spin_dialog_button=: 3 : 0
wd 'oledlg sx'
)
```

A press of the spinbutton, invokes the spinbutton handler: `spin_sx_button=: 3 :`

```
0
if. sysocx-:'spinup' do.
  PAYNDX=: 4|>:PAYNDX
else.
  PAYNDX=: 4|<:PAYNDX
end.
```

```
showpay''  
)
```

This handler checks the value of `sysocx` and adjusts the text field accordingly.

## Examples

### Example: Graph control

File: examples\ocx\misc\graph.js has a simple example using Pinnacle's GRAPH32.OCX control.

For a fuller example:load 'examples\ocx\graph\grafdemo.js'

### Example: TreeView

This example assumes you have the TreeView OCX installed and assorted bitmaps.

Run the script examples\ocx\misc\tree.js to create a treeview control. An imagelist control is loaded with various bitmaps which are then used by the treeview control.

### Example: Controlling Microsoft Word

It is possible to automate the word.basic object. This could be a useful way of solving some printing problems. The following loads Word, and reads in the J readme.wri file:load 'examples\ocx\misc\word.js'

### Example: Excel OLE Automation

```
wd 'pc excel'
```

parent to hold ole child

```
wd 'cc x oleautomation:excel.application'
```

create excel.application object

```
wd 'oleset x base visible 1'
```

set base visible property to 1

```
wd 'oleget x workbooks'
```

get workbooks object as temp

```
wd 'oleid x workbooks'
```

temp to permanent as workbooks

```
wd 'olemethod x workbooks open foo.xls'
```

open workbook foo.xls

```
wd 'olemethod x workbooks add'
```

run workbooks add method add method returns temp as the new workbook object

```
wd 'oleget x temp name'
```

get workbook name

```
wd 'olemethod x base quit'
```

quit The excel.application object is defined in the system registry to create a new instance (private copy) of excel. The excel.sheet object creates a new worksheet in a shared excel.

Run the script and then run excel". It is possible to access properties and methods directly, as well as run macros. The getobject verb shows how to support some of the simpler syntax of VB. Run excelquit" to close excel.

## Tutorial: J OLE Server for Excel

This tutorial has been created with Excel 2002 and J501. Expect different behavior with other versions!

It will help to have the Excel help file readily available.

### Introduction

The purpose of OLE Automation is to allow a client program to run functions in a server program, and the basic idea is pretty straightforward - simply load J from Excel, then send it the required J functions for execution. In practice is it helpful to create Excel macros that provide cover functions for the basic tasks such as loading J, reading cells for transmission to J and so on. Thus you typically program with a mixture of J functions and Excel macros.

Functions provided by an OLE Server are referred to as *methods*, see [J OLE Automation Server p220](#)

In Excel, you can enter these method names in upper or lowercase. When you enter names in Excel, it gives them its default capitalization. Here we use lowercase throughout.

### Troubleshooting

You are going to be working with both J and Excel sessions active. It will be helpful to close down other applications to minimize screen clutter.

As you use OLE, commands sent from Excel may change the active focus to J. To enter new commands in Excel, click on the Excel session to change the active focus back to Excel.

Most of the time when things go wrong, you can simply shut down J and Excel and start again. Sometimes, the J server has been loaded but is not visible. You can check this by pressing Ctrl+Alt+Del simultaneously, which brings up the list of current applications loaded. If necessary, select J and click End Task.

Sometimes when you edit Excel macros, Excel closes down J - it closes the OLE Automation object which may in turn cause J to close. You will then need to re-



open the J OLE Automation object. If J has closed and you try to run an OLE command, the error message is "Object variable not Set". This problem occurs only while you are developing Excel macros, and should not occur when your application is in use.

One of the "user-friendly" features of Excel is to change your entry in a cell if it thinks it may be incorrect. For example, "i.5" gets changed into "I.5". To get around this, enter more letters, then backspace and delete the extra entries, for example, instead of "i.5" try entering "ii.5".

### Tutorial

Start by unloading all applications, then loading Excel. Arrange the window so that it covers only about half the screen. Open a new workbook if none is shown.

Bring up Visual Basic (Alt-F11 or Tools|Macro|Visual Basic Editor), and insert a new module sheet (Insert|Module).

With the module sheet visible, select menu item Tools|References and check both J DLL Server and J EXE Server, and click OK. In practice you need only check the server that will be used.

In the module sheet, enter:

Public js As Object

Sub jopen()

Set js = CreateObject("jexeserver")

js.Quit

js.Do "0!:0<1!:45\$0"

End Sub

The function jopen will be used to load the JEXEServer. Note that you can only run this once - you will get an error at this point if you try to open the server twice.

In jopen:

The first statement declares the name js that will be used for the JEXEServer.

js.Quit ensures that when Excel is closed, the J server will automatically terminate.

js.Do "0!:0<1!:45\$0" loads the standard profile.

### **Loading J**

Next open up the Immediate window for experimentation (if not already open). To do so, select menu item View|Immediate Window. You can enter a series of commands in this Window - when you press Enter, Excel runs the command in the line where the cursor is.

To load J, enter:

```
jopen
```

Experiment with show:

```
js.show 0
```

this hides the window

```
js.show 1
```

this shows it again

This means: run the *show* method of js, i.e. of the JEXEServer, with the given argument.

The J OLE Automation Server should be visible. Arrange the windows so that both Excel and J are visible. Note that not only is the J Server visible, but if you click on it to give it focus then you have full access to the regular J development system.

Next set on logging - this tells J to display commands sent by Excel in the J window:

```
js.log 1
```

### **Sending commands to J**

The required function is do, which takes a J sentence as its argument. Note that

Excel strings are delimited by the double quote, so that J quotes can be entered as is, and need not be doubled. Try:

```
js.do "i.4 5"
```

```
js.do " 1!:1<PROFILE"
```

You should see the statements and results in the J window.

### **Retrieving values from J**

The function get retrieves a value from J, as a Variant datatype. Variants cannot be displayed directly in the Immediate window, but can be assigned to a worksheet range. For example:

Set value of x in J:

```
js.do "x=: i.4 5"
```

Retrieve value of x into Excel variant y:

```
js.get "x",y
```

Set value of y into the worksheet:

```
Worksheets("sheet1").Range("a1:e4")=y
```

Now switch to Sheet1 to see that the value of y has been written in.

### **Utilities**

Now let's take a look at the J OLE utilities in file  
system\examples\ole\excel\jsutil.txt. Copy and paste the contents of this file into  
your Excel module.

The utilities available are:

```
jdopen
```

```
open JDLLServer
```

jxopen

open JEXEServer

jcmd (string)

execute J command, return result as variant

jcmdc string,r,c,h,w

execute J command, store result in active sheet at row,col,height,width

jcmdr string,range

execute J command, store result in active sheet at range

jdo string

execute J command

jget(x)

get J noun x

jloadprofile

load standard J profile

jlog boolean

log on/off (EXE only)

jshow boolean

show on/off (EXE only)

You can customize these or add your own utilities.

### ***Loading J automatically***

In the Module, enter an auto\_open subroutine as follows:

```
Sub auto_open()  
jxopen  
jloadprofile  
jshow 1  
jlog 1  
End Sub
```

This sub will be run each time this workbook is opened. It opens the JEXEServer, shows the J session and logs commands sent from J.

Now check that auto\_open works correctly when you load the book. Switch back to Excel, save the book as test.xls and close Excel - note that the J session will close as well. Reload Excel, and open test.xls - you should see the J session again. Arrange the windows so that both Excel and J are visible.

### ***jcmd***

In Excel, switch to Sheet1 and in cell B3 enter:

=jcmd("/+2 3 5 7")

The statement should be executed in J, and the result (17) displayed in Excel.

Try:

In cell B5 enter: 12

In cell B6 enter: 15

In cell B7 enter: =jcmd(B5 & "\*" & B6)

B7 displays the result (180). Note that if you now change B5 or B6, then B7 will be recalculated.

In general, jcmd can be used for calculations which return a single value to be displayed in the current cell. The right argument is the sentence to be sent to J.

This method is really only suitable for simple calculations. Typically, you will want to run calculations that return a range of results to Excel and you set up such calculations by invoking an Excel macro explicitly, for example, by selecting Tools|Macro|Macros|Run or pressing an assigned hot-key.

### **jcmdc, jcmdr**

These utilities execute a J expression, displaying the result in a range in the active sheet. Function jcmdc specifies the range as 4 numbers: topleft row, column, number of rows, number of columns. Function jcmdr specifies the range in the traditional alphanumeric notation, for example: C6:E10.

We will create a macro run to test these and subsequent expressions. Switch to the module and enter:

```
Sub run()  
jcmdc "?3 4$10", 2, 3, 3, 4  
End Sub
```

Next, return to the worksheet, select Tools|Macro|Macros, highlight run and click Options. Enter Ctrl-r as the shortcut key and click OK. If J has been closed down, then either reload the spreadsheet or re-run auto\_open to load it again. Close the Macro dialog, switch to Sheet1 and press Ctrl-r. The macro should run and display the result. Press Ctrl-r again to re-run the macro.

### **jsetc, jsetr**

These utilities set values in J, from a range in the active sheet. As with jcmdc and jcmdr above, jsetc specifies the range as 4 numbers and function jsetr specifies the range in the traditional notation. Switch to the module and edit run to:

```
Sub run()  
jsetr "Y", "D3:F4"  
End Sub
```

Switch to Sheet1 and as before use Tools|Macro to select Ctrl-r as a shortcut key for the macro. In the worksheet, press Ctrl-r. Then click on the J session and display Y (these are random numbers so the exact values will likely differ):

```
Y  
+--+--+  
| 4 | 8 | 8 |  
+--+--+  
| 7 | 3 | 1 |  
+--+--+
```

Change run to:

```
Sub run()  
jsetc "Y", 2, 3, 3, 4
```

```
jcmdc "+^>Y", 7, 3, 3, 4  
End Sub
```

Switch to Sheet1, use Tools|Macro to select Ctrl-r as the shortcut key, then in the worksheet, press Ctrl-r. The macro will read the numbers in the upper range and display the sum scan in the lower range. Now if you change one of the numbers in the upper range, for example E2, press Ctrl-r to update the lower range.

## Tutorial: J OLE Client to Excel

This tutorial has been created with Excel 97 and J401. Expect different behavior with other versions!

It will help to have the Excel VBA help file VBAXL8.HLP (or similar) readily available.

When you create a new book in Excel, by default the book has 16 worksheets. Since we will experiment with adding worksheets under program control, we suggest that you change the default to 1 worksheet (use the Tools|Options|General dialog).

### Introduction

In theory, any Excel function can be called directly from J. In practice, some Excel functions have an unusual syntax that is either awkward or impossible to call from J, for example the ChartWizard method. However, you can always get around this by creating a corresponding Excel macro that you can call from J. Moreover it makes sense to use Excel macros anyway - there is no point in trying to duplicate in J a series of Excel function calls that could be just as easily, or more easily, programmed in Excel.

Thus you typically program with a mixture of J functions and Excel macros.

### Excel Hierarchy

The various parts of Excel such as the Workbooks, Worksheets and Charts (all known as *objects*) are organized in a hierarchy. Objects have *methods* (functions) and *properties* (variables). References to Excel objects, methods and properties must include their position in the hierarchy, for example:

```
Application.Workbooks("Book1").Worksheets("MySheet").ChartObjects.Item(1).Chart.PlotArea.Width
```

Now this naming convention gets a little tedious to enter, so Excel allows you to simplify it a little. For example, if MySheet happens to be the active sheet, you could instead use:

```
Activsheet.ChartObjects.Item(1).Chart.PlotArea.Width
```

J does not support this method of referencing names. Instead, for each reference you provide two names - the first being the position in the object hierarchy. Thus if the name `abc` represented `Activsheet.ChartObjects.Item(1).Chart.PlotArea` then the equivalent J reference would be `abc width`. How do you assign names in J to positions in the object hierarchy? To start off with, there are two reserved names. The name `base` represents the root of the hierarchy, equivalent to `Application` in Excel. Thus the following are equivalent:

```
base visible
```

J

```
Application.Visible
```

```
Excel
```



The name `temp` is assigned to the current position in the hierarchy. For example, if you have just created a new worksheet `Sheet1`, then the following are equivalent:

```
temp activate
```

J

```
Worksheets("Sheet1").Activate
```

Excel

Next, at any point, you can assign a name to the `temp` position. Thus if you assigned the name `sh1` to `temp` at this point, you could then use: `sh1 activate`

The idea is that you assign names to positions that you expect to revisit, while `temp` can be used for positions that you are just passing through.

Note that Excel names are not case-sensitive, but when programming, Excel automatically converts your entries to its standard capitalization. From J, you can use any case, and here we use lowercase throughout.

### Troubleshooting

You are going to be working with both J and Excel sessions active. It will be helpful to close down other applications to minimize screen clutter.

As you use OLE, commands sent from J may change the active focus to Excel. To enter new commands in J, click on the J session to change the active focus back to J.

If the OLE link goes wrong somewhere, you can simply close down the Excel session, and reset the J session. The J OLE interface uses the Window Driver, so you should enter `wd 'reset'` to reset it. Of course, normally you would shut down Excel and reset J under program control.

Sometimes you send a command to Excel that appears to hang up, while the Excel session flashes. This happens when Excel displays a dialog box that requires user intervention, for example an error message or a prompt to save changes on exit. In such cases, switch to Excel and respond to the dialog box before continuing.

At other times, Excel will hang up when it is waiting for user entry to be completed. For example, if you highlight a cell and start editing its contents, then switch to J and try an OLE command, the system will hang until you go back to Excel and complete the cell editing.

While Excel is fairly efficient at the tasks you are likely to use it for, you might inadvertently give it a task that takes a long time. For example, suppose you create a chart from data in a spreadsheet, then send a command from J to update that data. After each cell is updated, Excel will re-draw the chart - as many times as there are cells! While this happens, everything is locked up, and you will have to wait, or shut Excel down. (This particular problem is solved by erasing the chart before you update the data, then re-creating it after the update.)

## Tutorial

Start by unloading all applications, then loading J. Maximize the `ijx` window in the J session, then arrange the J session window so it covers only about half the screen.

### Opening up Excel

Create a parent to hold the Excel OLE Automation control:  
`wd 'pc xlauto'`

Create the Excel OLE Automation control. This may take a few seconds, because it loads Excel into memory (it will not be visible).  
`wd 'cc xl oleautomation:excel.application'`

All J OLE commands from now on will refer to the `xl` control. Note that the names `xlauto` and `xl` used here are not required – you can use your own names. However, the utilities included with J also use these names, so it is recommended that you stick with them. Note also that `xl` is short for Excel and not ``x`,`one``.

At this point, Excel has been loaded, but is not visible. Excel has a `Visible` property that can be set to display it. This property is part of the `Application` object, and hence the Excel call to use it would be:

`Application.Visible = 1`

In J, the `Application` object is named `base`, therefore to set it, enter:  
`wd 'oleaset xl base visible 1'`

This means: execute `oleaset` on control `xl`, setting the `visible` property of `base` to 1.

If Excel opens full screen, shrink it down so that both the J and Excel windows are visible.

Now Excel is visible, but has no workbook open. To create a new workbook in Excel, you use the `Add` method of the `Workbooks` object. Note that `Add` is a method of the `Workbooks` object (as well as several other objects), but is it not a method of the `Application` object - `Application.Add` will not work! Therefore the first step in J is to get access to the `Workbooks` object. To do so, enter:  
`wd 'oleget xl base workbooks'`

This command should complete successfully, but display no result. However, internally, J has assigned the `Workbooks` object to `temp`, and this can now be used to invoke the `Add` method:  
`wd 'olemethod xl temp add'`

This should have created a new workbook. Try entering it again to add another workbook:  
`wd 'olemethod xl temp add'`  
|domain error  
| wd'olemethod xl temp add'

This time you get a domain error - try: `wd 'qer'`. What happened is that the `temp` name really is temporary - it refers to the current position in the Excel hierarchy, which is constantly changing as you

move about Excel. In this case, when you added the new workbook, `temp` changed to that workbook - which does not have an `Add` method!

Therefore, in order to add another workbook, you have to assign `temp` to `Workbooks` again:

```
wd 'oleget xl base workbooks'  
wd 'olemethod xl temp add'
```

Of course, this quickly becomes tedious - therefore the proper treatment here is to assign a name to the `Workbooks` object, so that you can just use that name in future. To do so, use the `oleid` command:

```
wd 'oleget xl base workbooks'  
wd 'oleid xl wb'
```

Now you can use `wb` to create several books:

```
wd 'olemethod xl wb add'  
wd 'olemethod xl wb add'  
wd 'olemethod xl wb add'
```

## Closing Excel

Now lets try closing down Excel. The `Application` object in Excel has the `Quit` method to close down. `Quit` will prompt, should there be any unsaved changes. Try switching to Excel, then entering some values into one of the spreadsheets. Ensure that you have completed your entries (press `Enter` if Excel is waiting for you to complete the entry of a cell), then switch back to J and enter:  
`wd 'olemethod xl base quit'`

Excel will start flashing, and if you try to enter anything in the J window, it eventually displays a "Server Busy" dialog box. Click on Excel, and respond to the "Save changes in `Book'?" prompt. Eventually, Excel will close. You should now reset the J Window Driver with:  
`wd 'reset'`

## Utilities

This is a good time to look at the Excel OLE utilities, to do so enter:

```
load 'system\examples\ole\excel\xlutil.ijs'  
names ''
```

This defines several utilities, the main ones being:

`xlopen`

create Excel OLE automation object

`xlshow`

show/hide Excel OLE automation object

xlexit

exit Excel OLE automation object (saves)

xlget

cover for oleget - get object

xlset

cover for oleset - set object parameter

xlcmd

cover for olemethod - invoke method

xlid

cover for oleid - assign id to current position

xlread

read cell

xlreadr

read range

xlwrite

write cell

xlwriter

write range

xlsetchart

set chart range

The verb `xlopen` opens up Excel. It:

- creates the parent window `xlauto`
- creates the Excel OLE automation control `xl`
- names `wb` as the Workbooks object
- loads the macro file `examples\ole\excel\jmacros.xls` (which is hidden)

Try:`xlopen''`

Note that Excel is not shown, indeed you may want to use Excel without it ever being visible. To make it visible, enter:`xlshow''`

Verb `xlcmd` runs an OLE method. Since `wb` has been named in `xlopen`, it can be used directly. To add a workbook:`xlcmd 'wb add'`

Take a look at the workbook name:

```
xlget 'temp name'
Book1
```

Try changing the workbook name:

```
xlset 'temp name Mybook'
|domain error
| xlset'temp name Mybook'

wd 'qer'
ole - Workbook does not have writeable Name property : 12
```

What is happening is that in Excel, you can only change the name of a workbook by saving it. Thus, the following saves the workbook, and also renames it:

```
xlcmd 'temp saveas Mybook'
```

This may return -1, which really is the result from Excel!

(If you already have saved Mybook, Excel will prompt you to overwrite it.)

### Accessing the Worksheet

To access the worksheet, we first have to get the Worksheets object, which belongs to the workbook.

We will use the Worksheets object a few times, so will give it a name:

```
xlget 'temp worksheets'
xlid 'ws'
```

We can try adding new worksheets:

```
xlcmd 'ws add'
xlcmd 'ws add'
xlcmd 'ws add'
```

Next we access the first sheet using the Item method, and assign the name `sh1`:

```
xlget 'ws item sheet1'
xlid 'sh1'
xlget 'sh1 name'
Sheet1
```

Be careful to distinguish `sh1` which is the name used by J for a position in the Excel object hierarchy, from `Sheet1`, which is the name used by Excel for the current worksheet. You can change the worksheet name:

```
xlset 'sh1 name Mysheet'
```

If this worksheet is hidden behind another (which will be the case if you followed the above steps exactly) you can activate it with:

```
xlcmd 'sh1 activate'
```

Now lets try writing to a specific cell. In Excel you can use cell references of the form `2 3` or old-style alphanumeric references such as `B3`; the former are easier to program. First reference a cell, using the `Cells` property:

```
xlget 'sh1 cells 2 3'
```

Then set the value of `temp` as required. The new value should appear in the spreadsheet:

```
xlset 'temp value 123'
```

```
xlget 'temp value'
```

123

## Reading and Writing Ranges

In practice, you will typically want to read and write a range of cells. It would be tedious to do so one cell at a time; unfortunately, the form in which Excel reads and writes range data is not available to J. The solution is to use the utilities `xlreadr` (read range) and `xlwriter` (write range) that call appropriate macros from `jmacros.xls`. The right argument is the workbook, worksheet, topleft cell position and number of rows and columns. The left argument of `xlwriter` is the data to be written. Try:

```
(i.3 4) xlwriter 'mybook.xls mysheet 2 2'
```

Verb `xlreadr` returns data as a boxed array of character strings:

```
xlreadr 'mybook.xls mysheet 3 3 2 3'
+-----+
|5|6 |7 |
+-----+
|9|10|11|
+-----+
```

```
    ". &> xlreadr 'mybook.xls mysheet 3 3 2 3'  
5  6  7  
9 10 11
```

Finally, use `xlexit` to close Excel (you may be prompted to save):

```
xlexit''
```

### **Data passing**

Data parameters sent using these utilities are limited to 65K, which suffices for most purposes. The best way to pass data longer than this is via a temporary file. Thus J can write a file then send an OLE command to Excel to read it.

## **Labs**

[Overview p226](#)

[Lab Header p227](#)

[Lab Sections p228](#)

[Running Labs p229](#)

[Lab Author p230](#)

[Rich-Text p231](#)

[Program Access p232](#)



## Lab Overview

A J Lab file combines a text description and executable J code in a single file. The Lab system lets you step through the file, displaying the text and executing the code. The user is in a normal J session, and can try out other J expressions while running the lab.

A Lab is structured into one or more *chapters*, each containing one or more *sections*. The user can go through chapters in any order, but the sections within each chapter must be run sequentially.

Lab files are either ordinary text files with extension `.ijt`; or rich-text files with extension `.rtf`. Rich-text files permit greater control over text formatting, but work only in Windows 95/NT, and take more effort to set up.

You can create the ijt files using [Lab Author p230](#) or any text editor such as Notepad; and create the rtf files with word processors such as Word or WordPad, see [Rich-Text p231](#).

Lab files provided with J are stored under the subdirectory `system\extras\labs`. The directory `system\extras\labs\personal` is available for the user, though you can create labs in any directory - if so, you should update file `system\extras\labs\labdir.ijs`.

## Lab Header

A Lab file has a *header* defining various nouns, followed by one or more chapters that start with lines beginning `Lab Chapter` followed by the chapter name. If there is only one chapter, the line `Lab Chapter` need not be given.

Each chapter consists of sections that start with lines beginning `Lab Section` optionally followed by a section title. Where the section title is not given, it is assumed to be a continuation of the previous section.

### *Header*

The header is executed as a script in the `jlab` locale, and may be used to define the following nouns:

#### LABTITLE

Title (must be given first)

#### LABAUTHOR

Author (optional, may include address, email etc.)

#### LABCOMMENTS

Comments (optional)

#### LABERRORS

1=continue after errors in code (default 0)

#### LABFOCUS

Return focus to session after advancing the lab

#### LABNOSESSION

1=no output to J session (default 0)

## LABWIDTH

Text width (default 61)

## LABWINDOWED

1=run Lab with text in a window (default 1)

## LABWRAP

1=wrap text to LABWIDTH (default 1)

The title and author (if given) are shown when the lab is run.

Comments are not shown when the lab is started.

Continue after errors should be on if you intend to demonstrate errors in your J code, otherwise errors are signalled to the user. By default this is 0 (off).

Set no session output if you do not want to write to the J execution session. This may be appropriate for labs where there are no code examples; the labs may still contain code within keywords.

Run in window shows the lab text in a window. By default this is 1 (on); you may want to set this off for labs that create their own windows.

Text wrap and text width: by default text wrap is on, and means that any text output to the J execution window is wrapped to the specified text width (default 61).

The header may also contain any other required definitions, such as initialization code that is to be run at the beginning of each chapter. You can invoke any such definitions within `PREPARE` keywords in the first section of the chapter (see below). Since the header is defined in the jlab locale, subsequent references must use the full locale name.

## ***Chapters***

Chapters are delimited by a line beginning `Lab Chapter`, for example:

```
Lab Chapter Function Rank
```



## Lab Sections

### *Sections*

Sections are in two parts, either of which may be empty. The first part is the text to be displayed, ended by a line starting with `)`. Lines after the `)` are J sentences that are executed. If no `)` line is given, the section is assumed to be text only.

For example, the following is a lab section named "Numbers":

Lab Section Numbers

The integer function (i.) generates numbers:

)

i.10                      NB. first 10 numbers

i.4 3                      NB. first 12 numbers in a 4 by 3 table

The lab system responds advance by reading the next section, displaying the text, and executing the J sentences. All display is normal output to the active jx window, and the user has complete access to the J session.

### *Section Keywords*

Lines in the second part of a Lab Section (the J sentences to be executed) may begin with one of the keywords `SCRIPT`, `PREPARE` or `SOUND`.

- `SCRIPT` allows you to enter a character string that will be stored in the global variable `SCRIPT`, in the `jlab` locale. For example, this can be used to build up an example script.
- `PREPARE` allows you to enter sentences that will be run silently before the rest of the section is run. `>`.

The `SCRIPT` and `PREPARE` keywords are used to delimit text or sentences to be run before the rest of the section, and must occur at the beginning of the section. Any text on the same line as the keyword is ignored. For example, you can use these facilities to define the global `SCRIPT`, or load required code, and check whether it is OK to continue the lab.

A typical use of `PREPARE` is when your Lab creates Windows forms. To ensure that `wd` commands are sent to the selected form, start the code with a `pselect` command, for example:

```
PREPARE
wd 'pselect myform'
PREPARE
```

To prevent further execution of the section, signal an error. The utility `assert_lab_` may be used for this purpose; the left argument is the message to display when a 0 occurs in the right argument.

For example:

```
Lab Section Printing
The following prints the result:
)
PREPARE read in print fns -----
load 'print'
load 'myutils'
ERRORMSG=. 'Unable to load myutils',LF,LF,'Check they are
installed'
ERRORMSG assert_lab_ 3=nameclass <'myprintfn'
PREPARE -----
myprint RES
```

## Text Width

When using `ijt` (plain text) files, the recommended text width is 61 characters, which should display on all screens with typical screen fonts. You can create wider lines, but some users may have to scroll the screen in order to read them.

If `LABWRAP` is set on, text beginning at the left margin is automatically wrapped to width `LABWIDTH` when it is written to the J execution window. To avoid this, for example when including J code, indent text by one or more spaces.

## Ignored Lines

The system ignores any lines beginning `NB. ==.`

Files created with [Lab Author p230](#) have sections delimited with the line below, which has the text width used by the editor:

NB . =====

## Running Labs

A lab is invoked from menu item Studios|Labs, which runs the verb `lab` in the `j` locale. See [Lab Program Access p232](#).

The Lab Select form has an Intro to Labs button, which runs the lab `system\extras\labs\labintro.txt`, not listed with the other labs.

The Category selection box allows you to select labs in specific directories. These are defined in script `system\extras\labs\labdir.ijs`, which you can modify as needed to include your own lab directories. Subdirectories of `system\extras\lab` are automatically included in the list of categories.

You can step through a lab by selecting menu Studio|Advance, or pressing the corresponding shortcut key, when the J session has focus. A lab that creates a form may allow stepping through when the form has focus, by defining a handler such as:

```
myform_jctrl_fkey_z=: 3 : 'lab_j_ 0 [ wd''smselout;smfocus'''
```

You can also jump to other chapters of the lab from menu item Studios|Jump.



## Lab Author

The Lab Authoring system can be used to create lab files. Load it from menu item Studio/Author.

A lab is created as a header, plus lab chapters and sections. Each section is displayed in two panes - the top pane is the text, and the lower pane is the code. Either may be empty.

Press the Run button to run the code in any section. The Lab is run in the session window, i.e. LABWINDOWED is ignored. Any errors are displayed in a message box. The code is not run automatically as you navigate through the lab.

You can step through the lab by pressing the advance shortcut key when the Author form has focus. This simulates the normal running of the lab. Note that the Author form has to have the focus to advance.

To load and run the lab exactly as the user would run it, use the menu item File/Run Lab.

Note that if your Lab creates Windows forms, you need to ensure that wd commands are sent to the correct form. Start your code sections with appropriate PREPARE statements, for example:

```
PREPARE
wd 'psel myform'
PREPARE
```

The Wrap button wraps the text pane to the current width. Use Edit/UnWrap to undo; this is only available immediately after a wrap has been done. Otherwise, use Edit/Restore Section to restore the pane to its original state.

The width indicator shows the current default width. Only part of the width indicator will show if the choice of Font and Width makes it too wide to display in the window - in which case, resize the window, or reduce the Font or Width settings.

The Sounds items are enabled only when the lab has sounds. Check Enable to play the sounds when running each section. Use the Insert button to insert a sound in the code.

The menu items are mostly self-explanatory, for example:

Section/Restore Section

Restores the panes to the initial values when you moved to the section

Edit/Font...

Sets the font used in the two panes

Edit/Header...

Sets the lab header

## Rich-Text

The essential difference between the ijt and rtf file is that the text part of each section in the rtf file can contain formatting such as bold, superscript and color. When the lab is run, the formatted text is displayed in a Windows form with a rich-text control.

If you close the form, the lab can be run in the session in the usual way.

The facilities available in the Windows rich-text control are those of WordPad, and are a subset of those in Word and other word processors. Formatting instructions not supported by the rich-text control are ignored.

The layout of the rtf file is exactly the same as for the ijt file. Formatting for the non-text sections is ignored, and it may be helpful to apply distinctive formatting to each section to make it easy to read.

LABWIDTH and LABWRAP are ignored by the rich-text control, except when the text window is closed and the lab run in a J session.

## Lab Program Access

The verb `lab` in locale `j` runs the lab system. It is invoked by selecting the lab menu items in the Studio menu; and by pressing the lab advance shortcut when a lab has been loaded.

`lab` can also be called under program control as described below, and this can be helpful when creating and testing a lab. If you are using `lab` in this way, you may want to define `lab_z_ = . lab_j_` to remove the need for the locale reference.

The form is:

```
lab ''
```

Lab dialog for files in `system\extras\labs` directory. Invoked by menu item Studio|Labs...

```
lab 0 [,num]
```

Show next Lab Section (or section num). Invoked by menu item Studio|Advance, or by the corresponding shortcut key.

```
lab 1
```

Show jump dialog Invoked by menu item Studio|Jump...

```
lab 2
```

Run Lab Author Invoked by menu item Studio|Author...

```
lab 'directory'
```

Show lab dialog for files with extension `.ijt` in given directory.

```
lab 'filename' [;num]
```

Invoke lab on given file (at chapter `num`)

Note that you can run any file as a lab file, e.g.

```
lab 'system\extras\labs\labintro.txt'
```

# Index

## A

- a. [J 5.01 Release Highlights and Overview p1](#) • [Old Windows Release Notes p2](#) • [wd commands p130](#) • [Patterns p156](#) • [Utilities p158](#)
- a: [J 5.01 Release Highlights and Overview p1](#) • [Utilities p158](#)
- abbreviation(s) [Plot Colors p176](#)
- abort [winlib p78](#)
- aboutbox [J OLE/OCX Client p221](#)
- absence [Copyright p160](#)
- absolute [SQL Reserved Words p217](#)
- accelerator [Window Controls p113](#) • [Accelerator Keys p125](#)
- accurately [fontspec p132](#)
- acknowledge [Wait p111](#)
- activate(s) [Controlling p20](#) • [Hints p139](#) • [Mouse p140](#) • [Code p149](#) • [Tutorial: J OLE Client to Excel p224](#)
- activated [Overview p89](#)
- activex [Overview p219](#)
- actuarial [system\packages p23](#)
- administrator [Socket Driver p188](#)
- aggregation [Copyright p160](#)
- agree(s) [Copyright / Warranty / License p9](#)
- agreement(s) [Copyright / Warranty / License p9](#) • [Copyright p160](#)

aix [Products p6](#)

algebra [Old Windows Release Notes p2](#)

algorithm(s) [strings p71](#)

alphabetic [text p73](#) • [Overview p114](#) • [Code p149](#) • [Patterns p156](#)

alphabetically [Child Classes p118](#)

alphanumeric [Overview p114](#) • [wd commands p130](#) • [Patterns p156](#) • [Tutorial: J OLE Server for Excel p223](#) • [Tutorial: J OLE Client to Excel p224](#)

approximate(s) [SQL Elements p216](#)

approximately [Mapping Mode p134](#)

abortretryignore [wd commands p130](#)

arc(s) [graph p47](#) • [Definition Summaries p80](#) • [gl2 commands p131](#)

arccos [trig p74](#) • [Definition Summaries p80](#)

arccosh [trig p74](#) • [Definition Summaries p80](#)

arcsin [trig p74](#) • [Definition Summaries p80](#)

arcsinh [trig p74](#) • [Definition Summaries p80](#)

arctan [trig p74](#) • [Definition Summaries p80](#)

arctanh [trig p74](#) • [Definition Summaries p80](#)

argv [J 5.01 Release Highlights and Overview p1](#) • [sysenv p72](#) • [Definition Summaries p80](#)

argverb [J 5.01 Release Highlights and Overview p1](#)

arithmetic [dates p40](#) • [statfns p69](#) • [Definition Summaries p80](#)

ascii [J 5.01 Release Highlights and Overview p1](#) • [Old Windows Release Notes p2](#) • [format p46](#) • [stdlib p70](#) • [Definition Summaries p80](#) • [Printing p87](#) • [Richedit Control p119](#) • [Actions p166](#)

assert(s) [Old Windows Release Notes p2](#) • [stdlib p70](#) • [Definition Summaries p80](#) • [Lab Sections p228](#)

assertion(s) [stdlib p70](#) • [J Socket Protocol p190](#) • [SQL Reserved Words p217](#)

assignment(s) [J 5.01 Release Highlights and Overview p1](#) • [Find in Files p86](#) • [Regular Expression p155](#) • [Utilities p158](#)

assigns [wdhandler p107](#)

async [socket p67](#) • [Definition Summaries p80](#)

asynchronously [Socket Utilities p189](#)

authoring [Lab Author p230](#)

axe(s) [Plot Options p174](#) • [Plot Colors p176](#) • [Viewing p179](#)

## B

backslash [Patterns p156](#)

backspace [Old Windows Release Notes p2](#) • [Tutorial: J OLE Server for Excel p223](#)

barchart(s) [plot verb p170](#)

baserep [numeric p58](#) • [Definition Summaries p80](#)

bident(s) [J 5.01 Release Highlights and Overview p1](#)

binary [J 5.01 Release Highlights and Overview p1](#) • [Copyright p160](#) • [J Socket Protocol p190](#) • [SQL Elements p216](#)

binomial(s) [statdist p68](#) • [Definition Summaries p80](#)

binomialdist [statdist p68](#) • [Definition Summaries p80](#)

binomialprob [statdist p68](#) • [Definition Summaries p80](#)

binomialrand [statdist p68](#) • [Definition Summaries p80](#)

bitmap(s) [Mac J.402 Startup p11](#) • [system/packages p23](#) • [bmp p32](#) • [isigraph p48](#) • [Definition Summaries p80](#) • [Child Classes p118](#) • [Toolbar p122](#) • [Window Driver Command Reference p129](#) • [wd commands p130](#) • [gl2 commands p131](#) • [gl3 commands p135](#) • [OpenGL printing p136](#) • [Toolbar p147](#) • [Plot Commands p173](#) • [Examples p222](#)

bits [wd commands p130](#) • [gl2 commands p131](#)



bitsize [bmp p32](#)

bitpixel [wd commands p130](#) • [gl2 commands p131](#)

bitwise [dll p44](#) • [Definition Summaries p80](#)

bold [Printing p87](#) • [wd p104](#) • [Richedit Control p119](#) • [Fonts p124](#) • [fontspec p132](#) • [Plot Options p174](#) • [Rich-Text p231](#)

bool [Old Windows Release Notes p2](#) • [wd commands p130](#) • [gl2 commands p131](#) • [Calling DLLs p199](#) • [J OLE/OCX Client p221](#)

boolean(s) [isigraph p48](#) • [misc p55](#) • [stdlib p70](#) • [viewmat p76](#) • [Definition Summaries p80](#) • [Verbs p157](#) • [Plot Options p174](#) • [Java jserver class p182](#) • [Calling DLLs p199](#) • [Tutorial: J OLE Server for Excel p223](#)

box [Old Windows Release Notes p2](#) • [About J p5](#) • [J User License Order Form p7](#) • [Copyright / Warranty / License p9](#) • [Mac J.402 Startup p11](#) • [Script Windows p16](#) • [misc p55](#) • [stdlib p70](#) • [winlib p78](#) • [write p79](#) • [Definition Summaries p80](#) • [Printing p87](#) • [Project Conventions p100](#) • [wd p104](#) • [Window Forms p105](#) • [Wait p111](#) • [Parent Windows p115](#) • [Child Classes p118](#) • [Common Dialog Boxes p123](#) • [wd commands p130](#) • [gl2 commands p131](#) • [gl3 commands p135](#) • [Mouse p140](#) • [Toolbar p147](#) • [Statusbar p148](#) • [Verbs p157](#) • [Plot Options p174](#) • [Plot Colors p176](#) • [Examples p197](#) • [Data Driver p208](#) • [Listing the Data Sources p209](#) • [J OLE/OCX Client p221](#) • [Tutorial: J OLE Client to Excel p224](#) • [Running Labs p229](#) • [Lab Author p230](#)

boxed [J 5.01 Release Highlights and Overview p1](#) · [Old Windows Release Notes p2](#) · [Mac J.402 Startup p11](#) · [csv p39](#) · [dir p43](#) · [files p45](#) · [format p46](#) · [misc p55](#) · [myutil p56](#) · [stdlib p70](#) · [strings p71](#) · [text p73](#) · [validate p75](#) · [winlib p78](#) · [Definition Summaries p80](#) · [Component Files p84](#) · [Watch p93](#) · [wdhandler p107](#) · [gl3 commands p135](#) · [Verbs p157](#) · [pd verb p169](#) · [Plot Options p174](#) · [Plot Data p175](#) · [Socket Utilities p189](#) · [J Socket Protocol p190](#) · [Calling DLLs p199](#) · [Tutorial: J OLE Client to Excel p224](#)

boxes [Old Windows Release Notes p2](#) · [Status Bar p19](#) · [Find in Files p86](#) · [Wait p111](#) · [Window Controls p113](#) · [Child Controls p117](#) · [Child Classes p118](#) · [Common Dialog Boxes p123](#) · [wd commands p130](#) · [fontspec p132](#) · [isigraph events p133](#) · [Verbs p157](#) · [Plot Options p174](#)

boxopen [stdlib p70](#) · [Definition Summaries p80](#)

break(s) [J 5.01 Release Highlights and Overview p1](#) · [scriptdoc utility p31](#) · [winlib p78](#) · [Definition Summaries p80](#) · [gl2 commands p131](#) · [J Socket Protocol p190](#) · [J OLE Automation Server p220](#)

brk [J Socket Protocol p190](#)

byte(s) [Old Windows Release Notes p2](#) · [misc p55](#) · [socket p67](#) · [Definition Summaries p80](#) · [Component Files p84](#) · [gl2 commands p131](#) · [Java jserver class p182](#) · [J Socket Protocol p190](#) · [Calling DLLs p199](#) · [Memory Management p201](#)

## C

C. [J 5.01 Release Highlights and Overview p1](#)

calendar [scriptdoc utility p31](#) · [dates p40](#) · [Definition Summaries p80](#)

callback(s) [Old Windows Release Notes p2](#) · [dll p44](#) · [Definition Summaries p80](#) · [gl3 commands p135](#)

caret [J 5.01 Release Highlights and Overview p1](#) • [Find in Files p86](#) • [gl2 commands p131](#) • [Hints p139](#) • [Code p149](#)

categories [Running Labs p229](#)

category [Old Windows Release Notes p2](#) • [wd commands p130](#) • [Running Labs p229](#)

catenated [Old Windows Release Notes p2](#) • [Utilities p158](#) • [Calling DLLs p199](#)

catenating [Utilities p158](#)

cauchy [statdist p68](#) • [Definition Summaries p80](#)

cauchyrand [statdist p68](#) • [Definition Summaries p80](#)

classpath [Java p180](#) • [Java jserver class p182](#) • [Java classpath p183](#) • [Jsoftware Java applets p184](#) • [Java applet security p186](#)

clip [pd verb p169](#) • [Plot Commands p173](#)

clipboard [Session Manager p14](#) • [format p46](#) • [winlib p78](#) • [Definition Summaries p80](#) • [wd commands p130](#) • [gl2 commands p131](#) • [gl3 commands p135](#) • [Hints p139](#) • [Keyboard p141](#) • [Methods p164](#) • [Plot Commands p173](#) • [DDE Conversations p194](#)

clipcopy [wd commands p130](#)

clipfmt [format p46](#) • [Definition Summaries p80](#) • [DDE Conversations p194](#)

clippaste [wd commands p130](#)

coclass [colib p33](#) • [Definition Summaries p80](#) • [Building Applications p99](#)

coclasspath [colib p33](#)

cocreate [colib p33](#) • [Definition Summaries p80](#)

cocurrent [colib p33](#) • [Definition Summaries p80](#)

codestroy [colib p33](#) • [Definition Summaries p80](#) • [Methods p164](#)

coerase [colib p33](#) • [Definition Summaries p80](#)

coextend [colib p33](#) • [Definition Summaries p80](#)

coinfo [coutil p38](#) • [Definition Summaries p80](#)

coinsert [J 5.01 Release Highlights and Overview p1](#) · [colib p33](#) · [Definition Summaries p80](#) · [gl2 commands p131](#)

compared [J 5.01 Release Highlights and Overview p1](#)

component(s) [system\packages p23](#) · [jfiles p49](#) · [keyfiles p52](#) · [Definition Summaries p80](#) · [Development Environment p82](#) · [Component Files p84](#) · [Keyed Files p85](#) · [Find in Files p86](#) · [Copyright p160](#) · [Overview p219](#)

coname(s) [colib p33](#) · [Definition Summaries p80](#)

conew [J 5.01 Release Highlights and Overview p1](#) · [colib p33](#) · [Definition Summaries p80](#) · [Classes p163](#) · [Plot Class p171](#)

config [Old Windows Release Notes p2](#) · [Directory Paths p12](#) · [system\extras p25](#) · [user p27](#) · [sysenv p72](#) · [Definition Summaries p80](#) · [Menu Commands p83](#)

conj [format p46](#) · [misc p55](#) · [stdlib p70](#)

conjunction(s) [J 5.01 Release Highlights and Overview p1](#) · [Old Windows Release Notes p2](#) · [Directory Paths p12](#) · [system\packages p23](#) · [scriptdoc utility p31](#) · [stdlib p70](#) · [strings p71](#) · [Definition Summaries p80](#) · [Overview p89](#) · [Stops p92](#) · [Locked Scripts p101](#)

conl [colib p33](#) · [Definition Summaries p80](#) · [Classes p163](#)

constants [winapi p77](#) · [Definition Summaries p80](#) · [wd commands p130](#) · [Java jserver class p182](#) · [Socket Utilities p189](#) · [J OLE/OCX Client p221](#)

control(s) [J 5.01 Release Highlights and Overview p1](#) · [Old Windows Release Notes p2](#) · [Session Manager p14](#) · [Script Windows p16](#) · [system\packages p23](#) · [bmp p32](#) · [debug p42](#) · [files p45](#) · [isigraph p48](#) · [socket p67](#) · [viewmat p76](#) · [winlib p78](#) · [Definition Summaries p80](#) · [Verbs p90](#) · [Overview p103](#) · [wd p104](#) · [Window Forms p105](#) · [Event Handlers p106](#) · [wdhandler p107](#) · [Other Message Handlers p110](#) · [Window Controls p113](#) · [Overview p114](#) · [Parent Windows p115](#) · [Location and Size p116](#) · [Child Controls p117](#) · [Child Classes p118](#) · [Richedit Control p119](#) · [Tab Control p121](#) · [Toolbar p122](#) · [Fonts p124](#) · [Tab and Cursor Keys p127](#) · [Ownerdraw p128](#) · [wd commands p130](#) · [gl2 commands p131](#) · [isigraph events p133](#) · [Mapping Mode p134](#) · [gl3 commands p135](#) · [OpenGL printing p136](#) · [Form Editor p137](#) (only first 40 listed)

controll [Session Manager p13](#)

## D

D. [Old Windows Release Notes p2](#) · [Input Log p17](#)

database(s) [Old Windows Release Notes p2](#) · [Overview p204](#) · [Connection & Statement Handles p207](#) · [Data Driver p208](#) · [Listing the Data Sources p209](#)

date(s) [J User License p3](#) · [J User License Order Form p7](#) · [Copyright / Warranty / License p9](#) · [Mac J.402 Startup p11](#) · [system\main p22](#) · [system\packages p23](#) · [scriptdoc utility p31](#) · [dates p40](#) · [dir p43](#) · [kfiles p53](#) · [misc p55](#) · [validate p75](#) · [Definition Summaries p80](#) · [Definitions by Script p81](#) · [Keyed Files p85](#) · [Project Conventions p100](#) · [Copyright p160](#) · [Selecting & reading data p212](#) · [SQL Elements p216](#) · [SQL Reserved Words p217](#)

datefmt [system\packages p23](#)

dbr [Old Windows Release Notes p2](#) • [debug p42](#) • [Definition Summaries p80](#) • [Overview p89](#) • [Verbs p90](#)

dde [system\packages p23](#) • [wd p104](#) • [System Events p112](#) • [wd commands p130](#) • [DDE p191](#) • [DDE Overview p192](#) • [Server and Client p193](#) • [DDE Conversations p194](#) • [Examples p197](#)

deal(s) [random p64](#) • [Definition Summaries p80](#) • [Component Files p84](#) • [Event Handlers p106](#)

debug [J 5.01 Release Highlights and Overview p1](#) • [Old Windows Release Notes p2](#) • [system\main p22](#) • [debug p42](#) • [winlib p78](#) • [Definition Summaries p80](#) • [Definitions by Script p81](#) • [Menu Commands p83](#) • [Debug p88](#) • [Overview p89](#) • [Verbs p90](#) • [Commands p91](#) • [Stops p92](#) • [Watch p93](#) • [Socket Utilities p189](#) • [J Socket Protocol p190](#) • [J OLE Automation Server p220](#)

debugger [Java p181](#)

debugging [debug p42](#) • [Definition Summaries p80](#) • [Copyright p160](#) • [Java examples p185](#)

decommitted [J 5.01 Release Highlights and Overview p1](#) • [Old Windows Release Notes p2](#)

decommitted [J 5.01 Release Highlights and Overview p1](#) • [Old Windows Release Notes p2](#) • [wd commands p130](#)

derivative(s) [Copyright p160](#)

derived [Copyright p160](#) • [Overview p162](#)

determinant(s) [Old Windows Release Notes p2](#) • [system\packages p23](#)

diagonal(s) [wd commands p130](#) • [gl2 commands p131](#)

dimension(s) [Old Windows Release Notes p2](#)

dir(s) [Script Libraries p21](#) • [system\main p22](#) • [dir p43](#) • [Definition Summaries p80](#) • [Definitions by Script p81](#) • [Utilities p158](#)

distribute(s) [Old Windows Release Notes p2](#) • [Copyright p160](#)

distributed	<a href="#">Old Windows Release Notes p2</a> • <a href="#">About J p5</a> • <a href="#">Products p6</a> • <a href="#">Mac J.402 Startup p11</a> • <a href="#">Overview p95</a> • <a href="#">fontspec p132</a> • <a href="#">Copyright p160</a> • <a href="#">Java p181</a> • <a href="#">Java jserver class p182</a> • <a href="#">Overview p204</a> • <a href="#">The SQL Language p205</a> • <a href="#">Data Driver p208</a> • <a href="#">J OLE/OCX Client p221</a>
distributing	<a href="#">system\main p22</a> • <a href="#">Copyright p160</a>
distribution(s)	<a href="#">Old Windows Release Notes p2</a> • <a href="#">system\packages p23</a> • <a href="#">statdist p68</a> • <a href="#">Definition Summaries p80</a> • <a href="#">Overview p95</a> • <a href="#">Copyright p160</a>
distributive	<a href="#">numeric p58</a>
div	<a href="#">Copyright p160</a>
doc(s)	<a href="#">Old Windows Release Notes p2</a> • <a href="#">Common Dialog Boxes p123</a> • <a href="#">wd commands p130</a>
document(s)	<a href="#">J 5.01 Release Highlights and Overview p1</a> • <a href="#">Old Windows Release Notes p2</a> • <a href="#">Session Manager p14</a> • <a href="#">Controlling p20</a> • <a href="#">scriptdoc utility p31</a> • <a href="#">gl2 commands p131</a> • <a href="#">gl3 commands p135</a> • <a href="#">Copyright p160</a> • <a href="#">J Socket Protocol p190</a>
documentation	<a href="#">J 5.01 Release Highlights and Overview p1</a> • <a href="#">Old Windows Release Notes p2</a> • <a href="#">Copyright / Warranty / License p9</a> • <a href="#">Mac J.402 Startup p11</a> • <a href="#">scriptdoc utility p31</a> • <a href="#">gl3 commands p135</a> • <a href="#">Socket Driver p188</a> • <a href="#">J OLE/OCX Client p221</a>
documented	<a href="#">Old Windows Release Notes p2</a> • <a href="#">Products p6</a> • <a href="#">Starting J p10</a> • <a href="#">scriptdoc utility p31</a> • <a href="#">Plot Commands p173</a> • <a href="#">Plot Options p174</a> • <a href="#">Calling DLLs p199</a>
documentation	<a href="#">J 5.01 Release Highlights and Overview p1</a> • <a href="#">Products p6</a>
dot(s)	<a href="#">gl2 commands p131</a> • <a href="#">Plot Types p172</a> • <a href="#">Socket Utilities p189</a>
doubled	<a href="#">Tutorial: J OLE Server for Excel p223</a>
downloadable	<a href="#">J 5.01 Release Highlights and Overview p1</a>
downloaded	<a href="#">Jsoftware Java applets p184</a> • <a href="#">Java applet security p186</a>

draw(s) [graph p47](#) · [isigraph p48](#) · [write p79](#) · [Definition Summaries p80](#)  
 · [gl2 commands p131](#) · [OpenGL printing p136](#) · [Mouse p140](#) ·  
[Overview p168](#) · [plot verb p170](#) · [Plot Class p171](#) · [Tutorial: J OLE Client to Excel p224](#)

drop(s) [Old Windows Release Notes p2](#) · [keyfiles p52](#) · [stdlib p70](#) ·  
[strings p71](#) · [Definition Summaries p80](#) · [Menus p126](#) · [wd commands p130](#) · [Creating a new file p214](#) · [SQL Statements p215](#) · [SQL Reserved Words p217](#)

dsn [dd p41](#) · [Data Driver p208](#) · [Data Source Connection p211](#)

duplex [gl2 commands p131](#)

duplicate(s) [Input Log p17](#) · [jfiles p49](#) · [validate p75](#) · [Definition Summaries p80](#) · [Component Files p84](#) · [Tutorial: J OLE Client to Excel p224](#)

duplicate [Component Files p84](#)

duplicating [Component Files p84](#)

duplication [Copyright / Warranty / License p9](#)

dyad(s) [Old Windows Release Notes p2](#) · [stdlib p70](#) · [Definition Summaries p80](#)

dyadic [dll p44](#)

## E

e. [colib p33](#) · [regex p65](#) · [stdlib p70](#) · [Definition Summaries p80](#)

E. [regex p65](#) · [Definition Summaries p80](#)

eav [stdlib p70](#) · [Definition Summaries p80](#) · [Child Classes p118](#) ·  
[wd commands p130](#)

editm [Old Windows Release Notes p2](#) · [Child Classes p118](#) · [wd commands p130](#)

eigenpicture(s) [Old Windows Release Notes p2](#)



eigenvalue(s)	<a href="#">Old Windows Release Notes p2</a> • <a href="#">system\packages p23</a>
empties	<a href="#">Component Files p84</a>
enclose	<a href="#">Utilities p158</a>
enclosed	<a href="#">Copyright / Warranty / License p9</a> • <a href="#">Richedit Control p119</a> • <a href="#">Calling DLLs p199</a>
enclosing	<a href="#">Patterns p156</a>
enclosure	<a href="#">J User License Order Form p7</a>
encoded	<a href="#">Locked Scripts p101</a>
eol	<a href="#">Utilities p158</a>
equation(s)	<a href="#">Old Windows Release Notes p2</a> • <a href="#">system\packages p23</a>
evaluated	<a href="#">Session Manager p14</a> • <a href="#">Script Windows p16</a> • <a href="#">Examples p197</a>
evaluation	<a href="#">DDE Overview p192</a>
executable(s)	<a href="#">J 5.01 Release Highlights and Overview p1</a> • <a href="#">Directory Layout p29</a> • <a href="#">Copyright p160</a> • <a href="#">Examples p197</a> • <a href="#">Overview p226</a>
executes	<a href="#">J 5.01 Release Highlights and Overview p1</a> • <a href="#">Old Windows Release Notes p2</a> • <a href="#">wd p104</a> • <a href="#">Event Handlers p106</a> • <a href="#">wdhandler p107</a> • <a href="#">wd commands p130</a> • <a href="#">J Socket Protocol p190</a> • <a href="#">Examples p197</a> • <a href="#">Overview p219</a>
executing	<a href="#">J 5.01 Release Highlights and Overview p1</a> • <a href="#">Products p6</a> • <a href="#">wd p104</a> • <a href="#">Overview p226</a> • <a href="#">Lab Sections p228</a>
expand(s)	<a href="#">format p46</a> • <a href="#">stdlib p70</a> • <a href="#">Definition Summaries p80</a>
expandby	<a href="#">format p46</a> • <a href="#">Definition Summaries p80</a>
exponent(s)	<a href="#">SQL Elements p216</a>
exponential(s)	<a href="#">statdist p68</a> • <a href="#">Definition Summaries p80</a>
exponentialrand	<a href="#">statdist p68</a> • <a href="#">Definition Summaries p80</a>

## F

f.	<a href="#">Locked Scripts p101</a>
----	-------------------------------------

[factor\(s\)](#) [system\packages p23](#) • [isigraph p48](#) • [SQL Elements p216](#)  
[factorization\(s\)](#) [Old Windows Release Notes p2](#) • [system\packages p23](#)  
[fermat](#) [system\packages p23](#)  
[float\(s\)](#) [dll p44](#) • [Definition Summaries p80](#) • [gl3 commands p135](#) •  
[Calling DLLs p199](#) • [Memory Management p201](#) • [SQL Elements p216](#) • [SQL Reserved Words p217](#)  
[fonts](#) [Old Windows Release Notes p2](#) • [Mac J.402 Startup p11](#) •  
[Printing p87](#) • [Window Controls p113](#) • [Richedit Control p119](#) •  
[Fonts p124](#) • [fontspec p132](#) • [Lab Sections p228](#)  
[fontsize](#) [Printing p87](#) • [J OLE/OCX Client p221](#)  
[fontspec](#) [Printing p87](#) • [Window Driver Command Reference p129](#) • [wd commands p130](#) • [gl2 commands p131](#) • [fontspec p132](#) • [gl3 commands p135](#)  
[fork\(s\)](#) [J 5.01 Release Highlights and Overview p1](#)  
[forum](#) [Old Windows Release Notes p2](#) • [Support and Questions p8](#) •  
[bmp p32](#)  
[foxpro](#) [Overview p204](#) • [Listing the Data Sources p209](#)  
[fractal\(s\)](#) [Old Windows Release Notes p2](#)

## G

[gamma](#) [statdist p68](#) • [Definition Summaries p80](#)  
[gcd](#) [system\packages p23](#)  
[geometric](#) [statfns p69](#) • [Definition Summaries p80](#)  
[gnu](#) [Copyright p160](#)  
[gnuplot](#) [system\packages p23](#)  
[goto](#) [SQL Reserved Words p217](#)  
[grammar\(s\)](#) [SQL Statements p215](#) • [SQL Reserved Words p217](#)  
[grammer](#) [Data Driver p208](#)

## H

handler(s)	<a href="#">winlib p78</a> · <a href="#">Definition Summaries p80</a> · <a href="#">Window Driver p102</a> · <a href="#">Window Forms p105</a> · <a href="#">Event Handlers p106</a> · <a href="#">wdhandler p107</a> · <a href="#">Form Locales p109</a> · <a href="#">Other Message Handlers p110</a> · <a href="#">Wait p111</a> · <a href="#">System Events p112</a> · <a href="#">wd commands p130</a> · <a href="#">gl2 commands p131</a> · <a href="#">gl3 commands p135</a> · <a href="#">OpenGL printing p136</a> · <a href="#">Hints p139</a> · <a href="#">Menu p146</a> · <a href="#">Code p149</a> · <a href="#">Classes p163</a> · <a href="#">Methods p164</a> · <a href="#">Viewing p179</a> · <a href="#">Socket Utilities p189</a> · <a href="#">DDE Overview p192</a> · <a href="#">Communication Protocol p196</a> · <a href="#">Examples p197</a> · <a href="#">J OLE/OCX Client p221</a> · <a href="#">Running Labs p229</a>
head(s)	<a href="#">Directory Paths p12</a> · <a href="#">stdlib p70</a> · <a href="#">Definition Summaries p80</a>
hex	<a href="#">convert p37</a> · <a href="#">format p46</a> · <a href="#">Definition Summaries p80</a> · <a href="#">Patterns p156</a> · <a href="#">J Socket Protocol p190</a>
hexdump	<a href="#">format p46</a> · <a href="#">Definition Summaries p80</a>
hyphen	<a href="#">Patterns p156</a>

## I

i.	<a href="#">J 5.01 Release Highlights and Overview p1</a> · <a href="#">Old Windows Release Notes p2</a> · <a href="#">debug p42</a> · <a href="#">format p46</a> · <a href="#">isigraph p48</a> · <a href="#">jfiles p49</a> · <a href="#">misc p55</a> · <a href="#">numeric p58</a> · <a href="#">regex p65</a> · <a href="#">stdlib p70</a> · <a href="#">Definition Summaries p80</a> · <a href="#">Component Files p84</a> · <a href="#">Printing p87</a> · <a href="#">Wait p111</a> · <a href="#">Overview p168</a> · <a href="#">Plot Class p171</a> · <a href="#">Plot Data p175</a> · <a href="#">J Socket Protocol p190</a> · <a href="#">DDE Overview p192</a> · <a href="#">Examples p197</a> · <a href="#">Calling J.DLL p202</a> · <a href="#">Tutorial: J OLE Server for Excel p223</a> · <a href="#">Tutorial: J OLE Client to Excel p224</a> · <a href="#">Lab Sections p228</a>
i:	<a href="#">Old Windows Release Notes p2</a> · <a href="#">Directory Paths p12</a>

[icon\(s\)](#) [J 5.01 Release Highlights and Overview p1](#) • [Old Windows Release Notes p2](#) • [Mac J.402 Startup p11](#) • [Child Classes p118](#) • [Ownerdraw p128](#) • [wd commands p130](#) • [Socket Driver p188](#) • [Installing ODBC p206](#)

[ide](#) [J 5.01 Release Highlights and Overview p1](#) • [Products p6](#) • [J Socket Protocol p190](#)

[image\(s\)](#) [Old Windows Release Notes p2](#) • [Child Classes p118](#) • [Toolbar p122](#) • [Window Driver Command Reference p129](#) • [wd commands p130](#) • [gl2 commands p131](#) • [gl3 commands p135](#) • [OpenGL printing p136](#)

[indexed](#) [J 5.01 Release Highlights and Overview p1](#) • [winlib p78](#)

[indexes](#) [system/packages p23](#)

[infinities](#) [Old Windows Release Notes p2](#)

[integral\(s\)](#) [dates p40](#)

[integrated](#) [Old Windows Release Notes p2](#)

[integration](#) [system/packages p23](#) • [Overview p219](#)

[invoke\(s\)](#) [Menu Commands p83](#) • [wdhandler p107](#) • [Other Message Handlers p110](#) • [Communication Protocol p196](#) • [J OLE/OCX Client p221](#) • [Tutorial: J OLE Client to Excel p224](#) • [Lab Header p227](#) • [Program Access p232](#)

[invoked](#) [Controlling p20](#) • [Window Forms p105](#) • [Event Handlers p106](#) • [Copyright p160](#) • [Running Labs p229](#) • [Program Access p232](#)

[invoking](#) [Tutorial: J OLE Server for Excel p223](#)

## J

[j.](#) [Old Windows Release Notes p2](#) • [Utilities p158](#)

[jacobi](#) [system/packages p23](#)

java [J 5.01 Release Highlights and Overview p1](#) • [system\extras p25](#) • [sysenv p72](#) • [Definition Summaries p80](#) • [wd commands p130](#) • [gl2 commands p131](#) • [Java p180](#) • [Java p181](#) • [Java jserver class p182](#) • [Java classpath p183](#) • [Jsoftware Java applets p184](#) • [Java examples p185](#) • [Java applet security p186](#) • [J Socket Protocol p190](#)

jsoftware [Old Windows Release Notes p2](#) • [J User License p3](#) • [About J p5](#) • [Products p6](#) • [J User License Order Form p7](#) • [Copyright / Warranty / License p9](#) • [Java p180](#) • [J Socket Protocol p190](#)

jul [J 5.01 Release Highlights and Overview p1](#) • [J User License p3](#) • [Products p6](#) • [J User License Order Form p7](#)

## K

keyboard [J 5.01 Release Highlights and Overview p1](#) • [Accelerator Keys p125](#) • [gl2 commands p131](#) • [Form Editor p137](#) • [Overview p138](#) • [Keyboard p141](#) • [Classes p163](#) • [J Socket Protocol p190](#)

keyfiles [system\packages p23](#) • [keyfiles p52](#) • [kfiles p53](#) • [Definition Summaries p80](#) • [Definitions by Script p81](#) • [Keyed Files p85](#)

keyword(s) [Old Windows Release Notes p2](#) • [Directory Paths p12](#) • [keyfiles p52](#) • [kfiles p53](#) • [Definition Summaries p80](#) • [Keyed Files p85](#) • [wd commands p130](#) • [Calling DLLs p199](#) • [SQL Reserved Words p217](#) • [Lab Header p227](#) • [Lab Sections p228](#)

kfiles [system\packages p23](#) • [kfiles p53](#) • [Definition Summaries p80](#) • [Definitions by Script p81](#)

## L

L: [stdlib p70](#) • [Definition Summaries p80](#)

languages	<a href="#">J 5.01 Release Highlights and Overview p1</a> • <a href="#">Old Windows Release Notes p2</a>
lapack	<a href="#">Old Windows Release Notes p2</a>
largest	<a href="#">gl3 commands p135</a>
latent	<a href="#">Old Windows Release Notes p2</a> • <a href="#">debug p42</a> • <a href="#">Definition Summaries p80</a> • <a href="#">Overview p89</a>
legendre	<a href="#">system/packages p23</a>
letter(s)	<a href="#">Mac J.402 Startup p11</a> • <a href="#">Controlling p20</a> • <a href="#">scriptdoc utility p31</a> • <a href="#">text p73</a> • <a href="#">Child Controls p117</a> • <a href="#">wd commands p130</a> • <a href="#">Patterns p156</a> • <a href="#">Verbs p157</a> • <a href="#">Tutorial: J OLE Server for Excel p223</a>
libraries	<a href="#">Building Applications p99</a>
library	<a href="#">J 5.01 Release Highlights and Overview p1</a> • <a href="#">Old Windows Release Notes p2</a> • <a href="#">Products p6</a> • <a href="#">Script Libraries p21</a> • <a href="#">system/main p22</a> • <a href="#">Script Library Overview p30</a> • <a href="#">colib p33</a> • <a href="#">stdlib p70</a> • <a href="#">winlib p78</a> • <a href="#">Overview p95</a> • <a href="#">Project File p96</a> • <a href="#">Project Manager Tabs p98</a> • <a href="#">Building Applications p99</a> • <a href="#">Copyright p160</a>
limit(s)	<a href="#">Old Windows Release Notes p2</a> • <a href="#">debug p42</a> • <a href="#">wd commands p130</a> • <a href="#">Socket Utilities p189</a>
linux	<a href="#">J 5.01 Release Highlights and Overview p1</a> • <a href="#">Products p6</a>
lipshutz	<a href="#">Old Windows Release Notes p2</a>
locale(s)	<a href="#">J 5.01 Release Highlights and Overview p1</a> • <a href="#">Old Windows Release Notes p2</a> • <a href="#">Directory Paths p12</a> • <a href="#">Menus p18</a> • <a href="#">system/main p22</a> • <a href="#">colib p33</a> • <a href="#">coutil p38</a> • <a href="#">dll p44</a> • <a href="#">pack p59</a> • <a href="#">plot p61</a> • <a href="#">stdlib p70</a> • <a href="#">winlib p78</a> • <a href="#">Definition Summaries p80</a> • <a href="#">Menu Commands p83</a> • <a href="#">Component Files p84</a> • <a href="#">Printing p87</a> • <a href="#">Overview p89</a> • <a href="#">Stops p92</a> • <a href="#">Project File p96</a> • <a href="#">Building Applications p99</a> • <a href="#">Window Driver p102</a> • <a href="#">wdhandler p107</a> • <a href="#">Form Locales p109</a> • <a href="#">gl2 commands p131</a> • <a href="#">Classes p163</a> • <a href="#">J Socket Protocol p190</a> • <a href="#">Lab Header p227</a> • <a href="#">Lab Sections p228</a> • <a href="#">Running Labs p229</a> • <a href="#">Program Access p232</a>

locals [debug p42](#) • [Definition Summaries p80](#) • [Verbs p90](#)

log(s) [Session Manager p13](#) • [Input Log p17](#) • [Menu Commands p83](#) • [wd commands p130](#) • [Plot Options p174](#) • [Java jserver class p182](#) • [J OLE Automation Server p220](#) • [Tutorial: J OLE Server for Excel p223](#)

lowercase [J 5.01 Release Highlights and Overview p1](#) • [wd p104](#) • [Child Classes p118](#) • [Patterns p156](#) • [Tutorial: J OLE Server for Excel p223](#) • [Tutorial: J OLE Client to Excel p224](#)

## M

macintosh [Products p6](#) • [Mac J.402 Startup p11](#) • [gl2 commands p131](#) • [Socket Driver p188](#)

mailto [Copyright / Warranty / License p9](#) • [Directory Paths p12](#)

mailto:info [About J p5](#)

mailto:sales [J User License p3](#)

mailto:salesinfo [About J p5](#)

mailto:tech [About J p5](#)

mantissa [SQL Elements p216](#)

math [system\packages p23](#)

mathematical [Old Windows Release Notes p2](#)

mathutil [system\packages p23](#)

matrices [system\packages p23](#) • [compare p36](#) • [dates p40](#)

matrix [Old Windows Release Notes p2](#) • [system\packages p23](#) • [bmp p32](#) • [convert p37](#) • [dates p40](#) • [dir p43](#) • [files p45](#) • [format p46](#) • [isigraph p48](#) • [misc p55](#) • [numeric p58](#) • [stdlib p70](#) • [strings p71](#) • [text p73](#) • [validate p75](#) • [viewmat p76](#) • [winlib p78](#) • [Definition Summaries p80](#) • [Common Dialog Boxes p123](#) • [gl3 commands p135](#) • [Properties p165](#) • [Plot Options p174](#) • [Plot Data p175](#) • [J Socket Protocol p190](#)

member(s) [Old Windows Release Notes p2](#)

memory [Old Windows Release Notes p2](#) · [dll p44](#) · [jmf p50](#) · [Definition Summaries p80](#) · [DLLs and Memory Management p198](#) · [Calling DLLs p199](#) · [Memory Management p201](#) · [Calling J.DLL p202](#) · [J OLE Automation Server p220](#) · [Tutorial: J OLE Client to Excel p224](#)

modal [Wait p111](#)

module(s) [Copyright p160](#) · [SQL Reserved Words p217](#) · [Tutorial: J OLE Server for Excel p223](#)

monadically [wd p104](#)

## N

NB. [J 5.01 Release Highlights and Overview p1](#) · [Old Windows Release Notes p2](#) · [scriptdoc utility p31](#) · [publish p63](#) · [Verbs p90](#) · [Project File p96](#) · [Locked Scripts p101](#) · [Entering Information p108](#) · [Menus p126](#) · [wd commands p130](#) · [gl2 commands p131](#) · [gl3 commands p135](#) · [OpenGL printing p136](#) · [Verbs p157](#) · [Utilities p158](#) · [pd verb p169](#) · [plot verb p170](#) · [Plot Class p171](#) · [Java applet security p186](#) · [Socket Utilities p189](#) · [J Socket Protocol p190](#) · [Calling J.DLL p202](#) · [J OLE/OCX Client p221](#) · [Lab Sections p228](#)

nested [J Socket Protocol p190](#)

nub [misc p55](#) · [Definition Summaries p80](#)

nubcount [misc p55](#) · [Definition Summaries p80](#)

null(s) [Old Windows Release Notes p2](#) · [gl2 commands p131](#) · [Patterns p156](#) · [Utilities p158](#) · [J Socket Protocol p190](#) · [Calling DLLs p199](#) · [SQL Statements p215](#) · [SQL Reserved Words p217](#)

numerical [Copyright p160](#)



## O

obverse(s)	<a href="#">J 5.01 Release Highlights and Overview p1</a>
odbc	<a href="#">Old Windows Release Notes p2</a> · <a href="#">Mac J.402 Startup p11</a> · <a href="#">dd p41</a> · <a href="#">Definition Summaries p80</a> · <a href="#">ODBC Data Driver p203</a> · <a href="#">Overview p204</a> · <a href="#">The SQL Language p205</a> · <a href="#">Installing ODBC p206</a> · <a href="#">Data Driver p208</a> · <a href="#">Listing the Data Sources p209</a> · <a href="#">ODBC error messages p210</a> · <a href="#">SQL Statements p215</a> · <a href="#">SQL Elements p216</a> · <a href="#">SQL Reserved Words p217</a>
ole	<a href="#">J 5.01 Release Highlights and Overview p1</a> · <a href="#">Old Windows Release Notes p2</a> · <a href="#">wd p104</a> · <a href="#">wd commands p130</a> · <a href="#">Java p181</a> · <a href="#">J Socket Protocol p190</a> · <a href="#">OLE &amp; OCX p218</a> · <a href="#">Overview p219</a> · <a href="#">J OLE Automation Server p220</a> · <a href="#">J OLE/OCX Client p221</a> · <a href="#">Examples p222</a> · <a href="#">Tutorial: J OLE Server for Excel p223</a> · <a href="#">Tutorial: J OLE Client to Excel p224</a>
operator(s)	<a href="#">SQL Elements p216</a>

## P

parentheses	<a href="#">Utilities p158</a>
parenthesis	<a href="#">Patterns p156</a>
parenthesized	<a href="#">myutil p56</a> · <a href="#">Definition Summaries p80</a>
partitioned	<a href="#">parts p60</a> · <a href="#">Definition Summaries p80</a>
partition(s)	<a href="#">system\main p22</a> · <a href="#">misc p55</a> · <a href="#">parts p60</a> · <a href="#">Definition Summaries p80</a>

pixel(s)	<a href="#">J 5.01 Release Highlights and Overview p1</a> • <a href="#">Controlling p20</a> • <a href="#">graph p47</a> • <a href="#">Definition Summaries p80</a> • <a href="#">wd p104</a> • <a href="#">Location and Size p116</a> • <a href="#">wd commands p130</a> • <a href="#">gl2 commands p131</a> • <a href="#">fontspec p132</a> • <a href="#">isigraph events p133</a> • <a href="#">Mapping Mode p134</a> • <a href="#">Plot Commands p173</a>
plots	<a href="#">Overview p168</a> • <a href="#">Plot Class p171</a> • <a href="#">Plot Commands p173</a> • <a href="#">Plot Options p174</a> • <a href="#">Plot Colors p176</a>
plotting	<a href="#">Overview p168</a> • <a href="#">Plot Options p174</a>
pocketpc	<a href="#">Products p6</a>
poisson	<a href="#">statdist p68</a> • <a href="#">Definition Summaries p80</a>
poissondist	<a href="#">statdist p68</a> • <a href="#">Definition Summaries p80</a>
poissonprob	<a href="#">statdist p68</a> • <a href="#">Definition Summaries p80</a>
poissonrand	<a href="#">statdist p68</a> • <a href="#">Definition Summaries p80</a>
polynomial(s)	<a href="#">system/packages p23</a>
posix	<a href="#">J 5.01 Release Highlights and Overview p1</a>
pousse	<a href="#">Old Windows Release Notes p2</a>
power(s)	<a href="#">Mac J.402 Startup p11</a> • <a href="#">Component Files p84</a> • <a href="#">Java p181</a> • <a href="#">Overview p219</a>
powerpc	<a href="#">Mac J.402 Startup p11</a>
prime(s)	<a href="#">system/packages p23</a>
primitive(s)	<a href="#">J 5.01 Release Highlights and Overview p1</a> • <a href="#">Old Windows Release Notes p2</a> • <a href="#">system/packages p23</a>
probability	<a href="#">statdist p68</a> • <a href="#">Definition Summaries p80</a>

## Q

q:	<a href="#">myutil p56</a> • <a href="#">Calling J.DLL p202</a>
quadratic	<a href="#">system/packages p23</a>

## R

r.	<a href="#">Tutorial: J OLE Server for Excel p223</a>
radian(s)	<a href="#">trig p74</a> · <a href="#">Definition Summaries p80</a>
random	<a href="#">system\packages p23</a> · <a href="#">numeric p58</a> · <a href="#">random p64</a> · <a href="#">statdist p68</a> · <a href="#">viewmat p76</a> · <a href="#">Definition Summaries p80</a> · <a href="#">Definitions by Script p81</a> · <a href="#">Tutorial: J OLE Server for Excel p223</a>
randomize	<a href="#">numeric p58</a> · <a href="#">random p64</a> · <a href="#">Definition Summaries p80</a>
rank(s)	<a href="#">Old Windows Release Notes p2</a> · <a href="#">format p46</a> · <a href="#">stdlib p70</a> · <a href="#">Window Driver Command Reference p129</a> · <a href="#">J Socket Protocol p190</a> · <a href="#">Calling J.DLL p202</a> · <a href="#">Lab Header p227</a>
raster	<a href="#">wd commands p130</a> · <a href="#">gl2 commands p131</a>
rasterization	<a href="#">gl2 commands p131</a>
ratio(s)	<a href="#">Mapping Mode p134</a> · <a href="#">Plot Options p174</a>
ravel(s)	<a href="#">gl2 commands p131</a>
ravelled	<a href="#">files p45</a>
razed	<a href="#">Verbs p157</a>
recurse	<a href="#">dir p43</a>
recursive	<a href="#">dir p43</a>
richedit	<a href="#">system\packages p23</a> · <a href="#">Window Controls p113</a> · <a href="#">Child Classes p118</a> · <a href="#">Richedit Control p119</a> · <a href="#">wd commands p130</a>
rotated	<a href="#">Viewing p179</a>
rotation(s)	<a href="#">Viewing p179</a>
rounded	<a href="#">graph p47</a> · <a href="#">Definition Summaries p80</a> · <a href="#">gl2 commands p131</a>
rounding	<a href="#">numeric p58</a> · <a href="#">gl2 commands p131</a>
roundint	<a href="#">numeric p58</a> · <a href="#">Definition Summaries p80</a>

## S

sandbox	<a href="#">wd commands p130</a> • <a href="#">Jsoftware Java applets p184</a> • <a href="#">Java applet security p186</a>
schaum	<a href="#">Old Windows Release Notes p2</a>
school(s)	<a href="#">J User License p3</a>
script(s)	<a href="#">J 5.01 Release Highlights and Overview p1</a> • <a href="#">Old Windows Release Notes p2</a> • <a href="#">Mac J.402 Startup p11</a> • <a href="#">Directory Paths p12</a> • <a href="#">Session Manager p13</a> • <a href="#">Session Manager p14</a> • <a href="#">Script Windows p16</a> • <a href="#">Input Log p17</a> • <a href="#">Menus p18</a> • <a href="#">Script Libraries p21</a> • <a href="#">system\main p22</a> • <a href="#">user p27</a> • <a href="#">Script Library Overview p30</a> • <a href="#">scriptdoc utility p31</a> • <a href="#">debug p42</a> • <a href="#">misc p55</a> • <a href="#">myutil p56</a> • <a href="#">publish p63</a> • <a href="#">stdlib p70</a> • <a href="#">Definition Summaries p80</a> • <a href="#">Definitions by Script p81</a> • <a href="#">Printing p87</a> • <a href="#">Overview p89</a> • <a href="#">Verbs p90</a> • <a href="#">Commands p91</a> • <a href="#">Overview p95</a> • <a href="#">Project File p96</a> • <a href="#">Project Manager Tabs p98</a> • <a href="#">Building Applications p99</a> • <a href="#">Project Conventions p100</a> • <a href="#">Locked Scripts p101</a> • <a href="#">Overview p103</a> • <a href="#">Entering Information p108</a> • <a href="#">Richedit Control p119</a> • <a href="#">Tab Control p121</a> • <a href="#">gl2 commands p131</a> • <a href="#">gl3 commands p135</a> • <a href="#">Overview p138</a> • <a href="#">Hints p139</a> (only first 40 listed)
scriptdoc	<a href="#">Script Libraries p21</a> • <a href="#">scriptdoc utility p31</a>
self	<a href="#">Lab Author p230</a>
semicolon(s)	<a href="#">format p46</a> • <a href="#">Printing p87</a> • <a href="#">wd p104</a> • <a href="#">wd commands p130</a> • <a href="#">pd verb p169</a> • <a href="#">plot verb p170</a> • <a href="#">Plot Options p174</a> • <a href="#">Data Driver p208</a>
shaped	<a href="#">graph p47</a> • <a href="#">Definition Summaries p80</a> • <a href="#">gl2 commands p131</a>
shapes	<a href="#">Plot Colors p176</a>
singleton(s)	<a href="#">format p46</a>
sinh	<a href="#">trig p74</a> • <a href="#">Definition Summaries p80</a>

[solaris Products p6](#)  
[sparse Old Windows Release Notes p2](#)  
[square\(s\) Old Windows Release Notes p2](#) · [system\packages p23](#) · [statfns p69](#) · [Definition Summaries p80](#) · [Copyright p160](#)  
[stdcall Calling DLLs p199](#)  
[stderr J 5.01 Release Highlights and Overview p1](#)  
[stdin J 5.01 Release Highlights and Overview p1](#)  
[stdout J 5.01 Release Highlights and Overview p1](#)

## T

[t. J Socket Protocol p190](#)  
[tabs convert p37](#) · [print p62](#) · [Project Manager p94](#) · [Project Manager Tabs p98](#) · [Child Classes p118](#) · [Tab Control p121](#)  
[tanh trig p74](#) · [Definition Summaries p80](#)  
[teach J 5.01 Release Highlights and Overview p1](#)  
[teacher\(s\) J User License p3](#)  
[tech J 5.01 Release Highlights and Overview p1](#) · [About J p5](#) · [Support and Questions p8](#) · [Form Editor p137](#) · [Overview p138](#) · [Tech Notes p153](#)  
[technical About J p5](#) · [Support and Questions p8](#) · [Copyright / Warranty / License p9](#)  
[technology Java p181](#)  
[technotes Overview p138](#)  
[telnet J 5.01 Release Highlights and Overview p1](#)  
[theory Old Windows Release Notes p2](#) · [Classes p163](#) · [Tutorial: J OLE Client to Excel p224](#)  
[theta Plot Options p174](#)  
[tie\(s\) gl2 commands p131](#)

[timer](#) [System Events p112](#) • [wd commands p130](#)  
[tolower](#) [stdlib p70](#) • [Definition Summaries p80](#)  
[toupper](#) [stdlib p70](#) • [Definition Summaries p80](#)  
[transform\(s\)](#) [J 5.01 Release Highlights and Overview p1](#) • [Old Windows Release Notes p2](#)  
[transformation\(s\)](#) [J 5.01 Release Highlights and Overview p1](#)  
[transforming](#) [Copyright p160](#)  
[translate\(s\)](#) [Copyright p160](#) • [Viewing p179](#) • [SQL Reserved Words p217](#)  
[translated](#) [Copyright p160](#) • [Viewing p179](#)  
[translation\(s\)](#) [Copyright p160](#) • [Viewing p179](#) • [SQL Reserved Words p217](#)  
[transpose\(s\)](#) [J OLE Automation Server p220](#)  
[transposed](#) [J OLE Automation Server p220](#)  
[treeview](#) [Examples p222](#)  
[trident\(s\)](#) [J 5.01 Release Highlights and Overview p1](#)  
[trigonometric](#) [system\main p22](#) • [trig p74](#)  
[troubleshooting](#) [Tutorial: J OLE Server for Excel p223](#) • [Tutorial: J OLE Client to Excel p224](#)

## U

[underscored](#) [Demo p159](#)  
[uppercase](#) [Child Controls p117](#) • [Child Classes p118](#) • [Patterns p156](#) • [Utilities p158](#) • [Plot Options p174](#)  
[url\(s\)](#) [Jsoftware Java applets p184](#)  
[utilities](#) [J 5.01 Release Highlights and Overview p1](#) • [Products p6](#)

## V

verb(s)      [J 5.01 Release Highlights and Overview p1](#) • [Old Windows Release Notes p2](#) • [Directory Paths p12](#) • [scriptdoc utility p31](#) • [bmp p32](#) • [colib p33](#) • [compare p36](#) • [convert p37](#) • [coutil p38](#) • [csv p39](#) • [dates p40](#) • [dd p41](#) • [debug p42](#) • [dir p43](#) • [dll p44](#) • [files p45](#) • [format p46](#) • [graph p47](#) • [isigraph p48](#) • [jfiles p49](#) • [jmf p50](#) • [jselect p51](#) • [keyfiles p52](#) • [kfiles p53](#) • [menu p54](#) • [misc p55](#) • [myutil p56](#) • [nfiles p57](#) • [numeric p58](#) • [pack p59](#) • [parts p60](#) • [plot p61](#) • [print p62](#) • [publish p63](#) • [random p64](#) • [regex p65](#) • [rgb p66](#) • [socket p67](#) • [statdist p68](#) • [statfns p69](#) (only first 40 listed)

## W

warranties      [Copyright / Warranty / License p9](#)

warranty      [General Information p4](#) • [Copyright / Warranty / License p9](#) • [Copyright p160](#)

wdhandler      [winlib p78](#) • [Definition Summaries p80](#) • [Window Driver p102](#) • [Overview p103](#) • [wdhandler p107](#) • [Other Message Handlers p110](#) • [isigraph events p133](#) • [Socket Utilities p189](#) • [J Socket Protocol p190](#)

winapi      [Old Windows Release Notes p2](#) • [winapi p77](#) • [Definition Summaries p80](#) • [Definitions by Script p81](#)

wince      [gl2 commands p131](#)

workbook(s)      [Examples p222](#) • [Tutorial: J OLE Server for Excel p223](#) • [Tutorial: J OLE Client to Excel p224](#)

## X

x. [J 5.01 Release Highlights and Overview p1](#) • [bmp p32](#) • [dates p40](#) • [debug p42](#) • [dir p43](#) • [files p45](#) • [format p46](#) • [isigraph p48](#) • [misc p55](#) • [numeric p58](#) • [random p64](#) • [socket p67](#) • [statfns p69](#) • [stdlib p70](#) • [strings p71](#) • [text p73](#) • [viewmat p76](#) • [winlib p78](#) • [Definition Summaries p80](#) • [Locked Scripts p101](#)

xls [Listing the Data Sources p209](#)

## Y

y. [J 5.01 Release Highlights and Overview p1](#) • [bmp p32](#) • [dates p40](#) • [debug p42](#) • [dir p43](#) • [files p45](#) • [format p46](#) • [isigraph p48](#) • [misc p55](#) • [numeric p58](#) • [random p64](#) • [socket p67](#) • [statfns p69](#) • [stdlib p70](#) • [strings p71](#) • [viewmat p76](#) • [winlib p78](#) • [Definition Summaries p80](#) • [Verbs p90](#) • [Locked Scripts p101](#) • [Examples p197](#)

year(s) [scriptdoc utility p31](#) • [dates p40](#) • [Definition Summaries p80](#) • [Copyright p160](#) • [SQL Elements p216](#) • [SQL Reserved Words p217](#)