



The tool of thought for expert programming

Dyalog APL RIDE Reference Guide

RIDE Version 2.0

Dyalog Limited

Minchens Court, Minchens Lane
Bramley, Hampshire
RG26 5BH
United Kingdom

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

Dyalog is a trademark of Dyalog Limited
Copyright © 1982-2016



*Dyalog is a trademark of Dyalog Limited
Copyright © 1982 – 2016 by Dyalog Limited.
All rights reserved.*

Version 2.0

Revision: 20160218_200

No part of this publication may be reproduced in any form by any means without the prior written permission of Dyalog Limited, Minchens Court, Minchens Lane, Bramley, Hampshire, RG26 5BH, United Kingdom.

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

SQAPL is copyright of Insight Systems ApS.

Array Editor is copyright of davidliebtag.com

UNIX® is a registered trademark in the United States and other countries, licensed exclusively through X/Open Company Limited.

Windows® is a registered trademark of Microsoft Corporation in the United States and other countries.

Oracle®, Javascript™ and Java™ are registered trademarks of Oracle and/or its affiliates.

Mac OS® and OS X® (operating system software) are trademarks of Apple Inc., registered in the U.S. and other countries.

All other trademarks and copyrights are acknowledged.

Contents

1	ABOUT THIS DOCUMENT	5
1.1	Audience	5
2	INTRODUCTION	6
3	INSTALLATION	7
3.1	Pre-requisites	7
3.2	Installation	7
3.2.1	Linux.....	8
3.2.2	Mac OS.....	8
3.2.3	Microsoft Windows.....	8
4	STARTING A DYALOG SESSION	9
4.1	The RIDE - Connect Dialog Box.....	9
4.1.1	Connect to an interpreter	10
4.1.2	Launch an interpreter	11
4.1.3	Listen for connections from interpreter	12
5	THE DYALOG DEVELOPMENT ENVIRONMENT	14
5.1	Menu Bar.....	14
5.1.1	File Menu	14
5.1.2	Edit Menu.....	14
5.1.3	View Menu	15
5.1.4	Actions Menu.....	16
5.1.5	Help Menu	16
5.2	Keyboard Key Mappings for APL Glyphs	16
5.2.1	Other Keyboard Options	17
5.3	Session User Interface.....	17
5.3.1	Caption.....	17
5.3.2	Language Bar.....	18
5.3.3	Session Window.....	18
6	INPUT WINDOWS	19
6.1	Session Window	19
6.2	Edit Window.....	19
6.2.1	Toolbar	20
6.2.2	Search and Replace	21
6.2.3	Exiting the Edit Window.....	22
6.3	Trace Window	22
6.3.1	Toolbar	23
6.3.2	Search	24
6.3.3	Exiting the Trace Window	24
7	WORKING IN A DYALOG SESSION	25
7.1	Keyboard Shortcuts and Command Codes.....	25
7.2	Navigating the Windows	25

7.3	Docking and Floating Windows	26
7.4	Entering APL Characters	26
7.5	Entering Expressions	27
7.5.1	Paired Enclosures	27
7.5.2	Autocomplete	27
7.5.3	Context-Sensitive Help	28
7.5.4	Syntax Colouring	28
7.6	Executing Expressions	29
7.6.1	Executing a New Expression	29
7.6.2	Re-executing a Previous Expression	29
7.6.3	Re-executing Multiple Previous Expressions	29
7.7	Suspending Execution	30
7.7.1	Breakpoints	30
7.7.2	Interrupts	31
7.8	Terminating a Dyalog Session Running Through the RIDE	31
8	RIDE-SPECIFIC LANGUAGE FEATURES	32
8.1	I-Beams	32
8.1.1	3500I : Send Text to RIDE-embedded Browser	32
8.1.2	3501I : Connected to the RIDE?	32
8.1.3	3502I : Enable RIDE in Run-time Interpreter	33
8.2	Configuration Parameters	33
8.2.1	RIDE_INIT	33
8.3	Unsupported Language Elements	34
8.3.1	Function Key Configuration	34
8.3.2	Underscored Characters	34
8.3.2.1	Underscored Characters in Window Captions	34
8.3.2.2	Underscored Characters in the Session	35
8.3.3	Operating System Terminal/Command Window Interaction	35
9	CUSTOMISING YOUR SESSION	36
9.1	View Menu	36
9.2	Preferences Dialog Box	36
9.2.1	Layout Tab	37
9.2.2	Shortcuts Tab	38
9.2.3	Code Tab	39
9.2.4	Colours Tab	40
9.2.5	Title Tab	41
9.2.6	Menu Tab	41
9.3	Environment Variables	42
APPENDIX A	DEFAULT KEYBOARD	43
APPENDIX B	KEYBOARD SHORTCUTS	44

1 About This Document

This document introduces the Remote Integrated Development Environment (RIDE). It describes the installation process and the RIDE's user interface (windows, menus, customisation options, keycode/keystroke mappings, etc.).

RIDE can be extensively customised; this document assumes that the default configuration is in use.

1.1 Audience

It is assumed that the reader has a working knowledge of Dyalog (for information on the resources available to help develop your Dyalog knowledge, see <http://www.dyalog.com/introduction.htm>).

2 Introduction



The use of the RIDE is subject to the conditions of your licence for Dyalog APL. The installation and use of the RIDE does not convey any additional rights to use any other Dyalog products. Specifically, although the interpreter can be configured to allow the RIDE to debug runtime executables, you should only do this if your licence also allows it.

On all platforms, Dyalog includes an Integrated Development Environment (IDE) to enable the interactive use of APL notation to explore data, discover algorithms and create solutions. Using the IDE, a user can create applications through experimentation and easily diagnose problems, resolve issues and resume work.

The RIDE is a cross-platform, graphical development environment capable of producing a rich user experience on a variety of platforms. It can be run on Mac OS, Microsoft Windows and Linux, and can be used from any one of these platforms as a front end for any supported Dyalog interpreter engine – including one running as a Service. The RIDE and connected Dyalog interpreter engines do not need to be running on the same device, the same type of device or even on the same continent.

The RIDE can be thought of as having two primary functions:

- Providing a user interface to an interpreter engine (local or remote).

The RIDE is the recommended IDE when running Dyalog on Mac OS or Linux. However, Dyalog's native IDE currently provides the richest environment for the development of APL applications for Microsoft Windows users. As the RIDE evolves and incorporates additional functionality this could change, but until then the Windows IDE is expected to remain the tool of choice for pure Windows development.

- Enabling/managing multiple connections to different interpreter engines (local or remote) from the same instance of the RIDE.

Multiple concurrent Dyalog Sessions can be run on the same instance of the RIDE, although each Dyalog Session can only be connected to a single instance of the RIDE. Connections can be made to interpreters installed on any supported operating system irrespective of the operating system that the RIDE is installed on; Dyalog does not need to be installed on the machine that the RIDE is installed on.

3 Installation

This chapter describes how to install the RIDE.

3.1 Pre-requisites

The RIDE 2.0 can only connect to a Dyalog interpreter that is version 14.1.



RIDE 2.0 is able to connect to a version 14.0 Dyalog interpreter, but some of the functionality described in this document could be restricted/unavailable.

The RIDE is supported on the following operating systems:

- Linux x86/x86_64 – the following distributions:
 - RHEL/CentOS 6 onwards
 - Debian 7 onwards
 - Ubuntu 14.04 onwards
 - OpenSUSE 13.1 onwards(distributions built on top of these will also work)
- Mac OS – OS X Yosemite onwards
- Microsoft Windows – Windows Vista onwards



For Mac OS, the hardware must have been introduced in 2010 or later.

3.2 Installation

Installation instructions are dependent on operating system.

If Dyalog is not installed on the machine that the RIDE is being installed on, then the APL385 font and keyboard mappings installed with the RIDE mean that they are available when running a Dyalog Session through the RIDE. However, to be able to enter APL glyphs outside a Dyalog Session (for example, in text files or emails) you will need to download and install the appropriate files (files and instructions are available from <http://www.dyalog.com/apl-font-keyboard.htm>, as is the Dyalog Unicode IME for Microsoft Windows).

3.2.1 Linux

The installation process for the RIDE is the same irrespective of whether it is installed as a stand-alone product or on a machine that already has Dyalog installed.

To install the RIDE:

1. Download the RIDE's **.deb** or **.rpm** file (whichever is appropriate for your Linux distribution) from my.dyalog.com. If your Linux distribution does not support either **.deb** or **.rpm** files, then please contact support@dyalog.com.
2. From the command line, use standard installation commands to install the package (in the default location).

The RIDE is now installed and ready to use. The RIDE icon (shortcut) is added to the desktop.

3.2.2 Mac OS

The RIDE is installed at the same time as Dyalog (see the *Dyalog for Mac OS Installation and Configuration Guide*); no further installation is required.

To install the RIDE as a stand-alone product:

1. Download the RIDE's **.pkg** file from my.dyalog.com.
2. Double-click on the RIDE's **.pkg** file.
The **RIDE Installer** window is displayed.
3. Follow the instructions in the **RIDE Installer** window to successful completion of the installation process.
The RIDE is now installed and ready to use. The RIDE icon is added to the **Applications** directory (accessed by selecting **Applications** from the **Go** menu in the **Finder** menu bar).

Starting the RIDE adds its icon to the dock. To keep the RIDE icon in the dock permanently, right-click on the icon and select **Options > Keep in Dock** from the drop-down list that appears.

3.2.3 Microsoft Windows

The installation process for the RIDE is the same irrespective of whether it is installed as a stand-alone product or on a machine that already has Dyalog installed.

To install the RIDE:

3. Download the RIDE's **.zip** file from my.dyalog.com.
4. Unzip the downloaded **.zip** file, placing the **setup_ride.exe** and **setup_ride.msi** files in the same location.
5. Double-click on the **setup_ride.exe** file.
The **RIDE Installation** window is displayed.
6. Follow the instructions in the **RIDE Installation** window to successful completion of the installation process.

The RIDE is now installed and ready to use. The RIDE icon (shortcut) is added to the desktop.

4 Starting a Dyalog Session



When running a Dyalog Session through the RIDE, that Session should only be accessed through the RIDE. One exception to this rule is when developing or running applications that are `□SM/□SR` based; access to the `□SM` window cannot be made through the RIDE.

When running a Dyalog Session through the RIDE, the Session can be:

- local to the machine on which the RIDE is running.
This requires Dyalog to be installed on the machine on which the RIDE is running.
- remote from the machine on which the RIDE is running.
The RIDE can start a Session using an interpreter installed on a remote machine irrespective of whether Dyalog is installed on the machine on which the RIDE is running. In this situation:
 - The operating system on which the remote interpreter is running is irrelevant – the instructions given in this chapter apply to the operating system on which the RIDE is running (the two operating systems do not have to be the same).
 - The remote machine does not need to have the RIDE installed but the Dyalog Session must be RIDE-enabled (see section 8.2.1).

Connections between the RIDE and Dyalog interpreters are initialised through the **RIDE - Connect** dialog box.

This chapter describes how to use the RIDE to run Dyalog Sessions, both local and remote.

4.1 The RIDE - Connect Dialog Box

When the RIDE is started, the **RIDE - Connect** dialog box is displayed (as shown in Figure 1).

The **RIDE - Connect** dialog box comprises three frames:

- the **Connect to an interpreter** frame – see section 4.1.1
- the **Launch an interpreter** frame – see section 4.1.2
- the **Listen for connections from interpreter** frame – see section 4.1.3

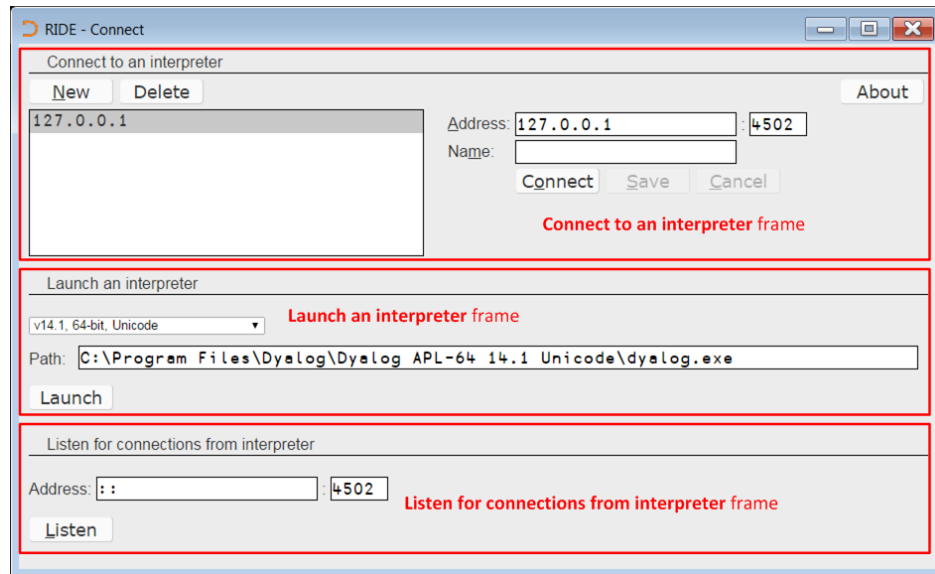


Figure 1. The RIDE - Connect dialog box (Microsoft Windows)

4.1.1 Connect to an interpreter

An interpreter client (local and/or remote) listens and the RIDE connects to it.

A specific Dyalog interpreter is sought by the RIDE for connection.

To start a Dyalog Session:

1. On the machine that the interpreter will run on, start a Dyalog Session. Optionally, specify the IP address and port that it will listen for RIDE connections on. If no port number is specified, then the port number defined in the RIDE_INIT environment variable is assumed (see section 8.2.1).

For example, if the RIDE is on a machine that has IP address 10.0.38.1 and will connect through port 4502, then enter the following in a command window/at the command prompt:

- on AIX:


```
> RIDE_INIT="SERVE:10.0.38.1:4502"
/opt/mdyalog/14.1/64/unicode/p7/dyalog
```
- on Linux:


```
> RIDE_INIT="SERVE:10.0.38.1:4502" dyalog
```
- on Mac OS:


```
> RIDE_INIT="SERVE:10.0.38.1:4502" /Dyalog/
Dyalog-14.1.app/Contents/Resources/Dyalog/map1
```
- on Microsoft Windows:


```
> cd C:\Program Files\Dyalog\Dyalog APL-64
14.1 Unicode
> set RIDE_INIT=SERVE:10.0.38.1:4502
> dyalog
```

alternatively, create a shortcut with the appropriate settings:

- a. Select the appropriate Dyalog installation and create a shortcut to it.
- b. Right-click on the shortcut icon and select **Properties** from the context menu that is displayed. The **Properties** dialog box is displayed.

- c. In the **Shortcut** tab, go to the **Target** field and:
 - o place " marks around the path
 - o append `RIDE_INIT=CONNECT:10.0.38.1:4502`
 For example: "C:\Program Files\Dyalog\Dyalog APL-64 14.1 Unicode\dyalog.exe"
`RIDE_INIT=CONNECT:10.0.38.1:4502`
- d. Click **OK**

The Dyalog Session starts.

2. On the machine that the RIDE is running on (localhost):
 - a. Open the **RIDE - Connect** dialog box.
 - b. In the **Connect to an interpreter** frame, select the interpreter to connect to in one of the following ways:
 - In the **Address** field , enter the IP address/unique DNS name of the machine that the interpreter is running on and set the port number to match that specified when starting the interpreter.
 - Select a previously-saved connection from the list of favourites.
 - c. Click **Connect**.

The Dyalog Session starts.

To add a new name/address to the list of favourites:

1. In the **Connect to an interpreter** frame, click **New**.
 2. Enter the IP address and/or DNS name in the **Address** field.
 3. Optionally, enter a port number after the address. If no port number is specified, then the port number defined in the `RIDE_INIT` environment variable is assumed (see section 8.2.1).
 4. Optionally, enter a name for this machine in the **Name** field.
 5. Click **Save**.
- The name (if defined) and address of the specified machine are added to the list of favourites.

To remove a name/address from the list of favourites:

1. In the **Connect to an interpreter** frame, select the name/address to be deleted in the list of favourites.
 2. Click **Delete**.
- The selected name/address are deleted from the list of favourites.

4.1.2 Launch an interpreter

The RIDE initiates and connects with an interpreter client (local only)

The drop-down list comprises all versions of Dyalog that are installed on the machine that the RIDE is running on; versions that are not supported by the RIDE are listed but not enabled.

If the path and/or name of the interpreter have been amended from the default installation values, then they might not appear in the drop-down list. An interpreter in this situation can be chosen by selecting *Other...* in the drop-down list and entering its full path in the **Path** field. The value of the **Path** field is remembered across invocations of the RIDE.

To start a Dyalog Session:

1. Open the **RIDE - Connect** dialog box.
2. In the **Launch an interpreter** frame, select the interpreter to run the Dyalog Session on from the drop-down list.
The path to the interpreter selected in the drop-down list is displayed in the **Path** field.
3. Click **Launch**.
The Dyalog Session starts.



Selecting **New Session** in the **File** menu (see section 5.1.1) launches another instance of the interpreter whose path is specified in the **Path** field.

4.1.3 Listen for connections from interpreter

The RIDE listens and interpreter clients (local and/or remote) connect to it

The RIDE listens on the local machine to connections made by Dyalog interpreters. The **Address** field can be used to restrict the sockets on which the RIDE listens (the address must be local to the machine that RIDE is running on).

The RIDE can listen using either IPv4 or IPv6 communications protocol; the default value of `::` listens using both IPv4 and IPv6 – if the machine that the RIDE is running on does not have an IPv6 stack then this should be changed to `0.0.0.0` instead.

To start a Dyalog Session:

1. On the machine that the RIDE is running on (localhost):
 - a. Open the **RIDE - Connect** dialog box.
 - b. In the **Listen for connections from interpreter** frame, optionally enter an IP address and/or change the default port number to restrict the sockets on which the RIDE listens (the address must be local to the machine that RIDE is running on). By default, the RIDE listens on port 4502.
 - c. Click **Listen**.
The **Waiting for connection...** dialog box is displayed.
2. On the machine that the interpreter will run on, start a Dyalog Session from the command prompt. When doing this, an appropriate IP address for the machine that the RIDE is running on and the same port number as the RIDE is listening on must be specified as connection properties.

For example, if the RIDE is running on a machine that has IP address 10.0.38.1 and is listening on port 4502, then enter the following in a command window/at the command prompt:

- on AIX:

```
> RIDE_INIT="CONNECT:10.0.38.1:4502"
/opt/mdyalog/14.1/64/unicode/p7/dyalog
```
- on Linux:

```
> RIDE_INIT="CONNECT:10.0.38.1:4502" dyalog
```
- on Mac OS:

```
> RIDE_INIT="CONNECT:10.0.38.1:4502" /Dyalog/
Dyalog-14.1.app/Contents/Resources/Dyalog/map1
```

- on Microsoft Windows:

```
> cd C:\Program Files\Dyalog\Dyalog APL-64
  14.1 Unicode
> set RIDE_INIT=CONNECT:10.0.38.1:4502
> dyalog
```

The Dyalog Session starts.

The RIDE will connect to the new Dyalog Session and remain connected unless the Dyalog Session is terminated.



On Microsoft Windows, an alternative to using the command window is to create a shortcut with the appropriate settings.

To configure the shortcut:

1. Select the appropriate Dyalog installation and create a shortcut to it.
2. Right-click on the shortcut icon and select **Properties** from the context menu that is displayed.

The **Properties** dialog box is displayed.

3. In the **Shortcut** tab, go to the **Target** field and:

- place " marks around the path
- append `RIDE_INIT=CONNECT:10.0.38.1:4502`

For example: "C:\Program Files\Dyalog\Dyalog APL-64 14.1 Unicode\dyalog.exe"
`RIDE_INIT=CONNECT:10.0.38.1:4502`

4. Click **OK**.
-

5 The Dyalog Development Environment

When a Dyalog Session is started through the RIDE, the Dyalog development environment is displayed. This means that:

- the appropriate menu bar menu options are displayed (see section 5.1)
- the keyboard key mappings for APL glyphs are enabled (see section 5.2)
- the Dyalog Session user interface is displayed (see section 5.3)

5.1 Menu Bar



The menu bar menu options on Mac OS are different to those on Microsoft Windows and Linux. This section details the options for Microsoft Windows and Linux; for the Mac OS options see the *Dyalog for Mac OS User Guide*.

The menu bar menu options are shown in Figure 2 – this section details each of the items in these menus.

File Edit View Actions Help

Figure 2. Menu bar menu options

The menu names in the menu bar and the options under each menu can be customised (see section 9.2.6).

5.1.1 File Menu

The options available under the **File** menu are detailed in Table 1. These control the RIDE-enabled APL process connections (both on local and remote machines).

Table 1. File menu options

Item	Description
New Session	Starts a new Dyalog Session (a new instance of the interpreter).
Connect...	Opens the RIDE - Connect dialog box (see chapter 4).
Quit	Terminates the Dyalog Session (see section 7.8).

5.1.2 Edit Menu

The options available under the **Edit** menu are detailed in Table 2. These assist with manipulating text within (and between) windows.

Table 2. Edit menu options

Item	Description
Cut	Deletes the selected text from the active window and places it on the clipboard.
Copy	Copies the selected text to the clipboard.
Paste	Pastes the text contents of the clipboard into the active window at the current location.
Undo	Reverses the previous action
Redo	Reverses the effect of the previous Undo
Preferences	Opens the Preferences dialog box (see section 9.2).

A context menu comprising the **Cut**, **Copy**, **Paste**, **Undo** and **Redo** options from the **Edit** menu is available in the **Session** window, all **Edit** windows and the **Trace** window.

5.1.3 View Menu

The options available under the **View** menu are detailed in Table 3. These enable the appearance of the Dyalog Session to be changed.

Table 3. View menu options

Item	Description
Show Language Bar	Toggles display of the language bar (see section 5.3.2) at the top of the Session window.
Float New Editors	Toggles whether new Edit or Trace windows are docked or floating (see section 7.3). Does not affect windows that are already open.
Editors on Top	Toggles whether Edit windows are always displayed on top of the Session window. Only relevant when the View > Float New Editors menu option is selected.
Line Wrapping in Session	Toggles use of automatic line wrapping (that is, whether text that exceeds the width of the window wraps to the next line or extends the line with a scroll bar) in the Session window.
Increase Font Size	Increases the size of the font in all the windows
Decrease Font Size	Decreases the size of the font in all the windows
Reset Font Size	Sets the size of the font in all the windows to its default value.
Theme	Influences the overall style of the user interface. Select from the drop-down list. The initial setting is: <ul style="list-style-type: none"> on Linux: <i>Classic</i> on Mac OS: <i>Cupertino</i> on Microsoft Windows: <i>Redmond</i>

5.1.4 Actions Menu

The options available under the **Actions** menu are detailed in Table 4. These allow currently-running APL code to be interrupted with trappable events.

Table 4. *Actions menu options*

Item	Description
Weak Interrupt	Interrupts the programme currently being run at the end of the current line.
Strong Interrupt	Interrupts the programme currently being run as soon as possible, even if this interrupts the line of code currently being executed.

5.1.5 Help Menu

The options available under the **Help** menu are detailed in Table 5. These provide access to the Dyalog documentation, website, forum and update portal.

Table 5. *Help menu options*

Item	Description
About	Displays the About dialog box, which provides details of the current Dyalog installation.
Dyalog Help	Opens your default web browser on the welcome page of Dyalog's online help (http://help.dyalog.com).
Documentation Centre	Opens your default web browser on the Documentation Centre page of Dyalog Ltd's website (http://www.dyalog.com/documentation.htm).
Dyalog Website	Opens your default web browser on the home page of Dyalog Ltd's website (http://www.dyalog.com).
MyDyalog	Opens your default web browser on the login page for MyDyalog, the customer portal for updates of Dyalog (https://my.dyalog.com).
Dyalog Forum	Opens your default web browser on the main page of Dyalog Ltd's forum (http://www.dyalog.com/forum).

5.2 Keyboard Key Mappings for APL Glyphs

A set of keyboard key mappings for APL glyphs is installed with the RIDE; these are shown in Appendix A. When the RIDE is the active application, these key mappings are automatically enabled. The RIDE attempts to identify a user's locale and use the appropriate keyboard; if the locale cannot be identified or the locale-specific keyboard has not been configured, then the default configuration is used (US keyboard).

Using the default keyboard, APL glyphs are entered by pressing the prefix key followed by either the appropriate key or the SHIFT key with the appropriate key. The prefix key and key mappings can be customised (see section 9.2.1).

5.2.1 Other Keyboard Options

The default keyboard can be replaced with a locale-specific keyboard in the Session – a locale-specific keyboard also allows Dyalog glyphs to be entered in other applications (for example, email) when it is selected.

Information on installing and enabling a locale-specific keyboard and the requisite downloadable files are available at <http://www.dyalog.com/apl-font-keyboard.htm>.



If you have the Dyalog Unicode IME installed, then the RIDE activates it by default. It can be disabled by unchecking the **Select Dyalog IME as the active keyboard** checkbox in the **Layout** tab of the **Preferences** dialog box (see section 9.2.1).

If Dyalog is not installed on the machine that the RIDE is running on, then the Dyalog Unicode IME can be downloaded and installed from <http://www.dyalog.com/apl-font-keyboard.htm>.



Most Linux distributions released after mid-2012 support Dyalog glyphs by default, for example, openSUSE 12.2, Ubuntu 12.10 and Fedora 17. For more information, see the *Dyalog for UNIX Installation and Configuration Guide*.

5.3 Session User Interface

The Dyalog Session user interface when running through the RIDE includes three elements – a caption, a language bar and a **Session** window (as shown in Figure 3).

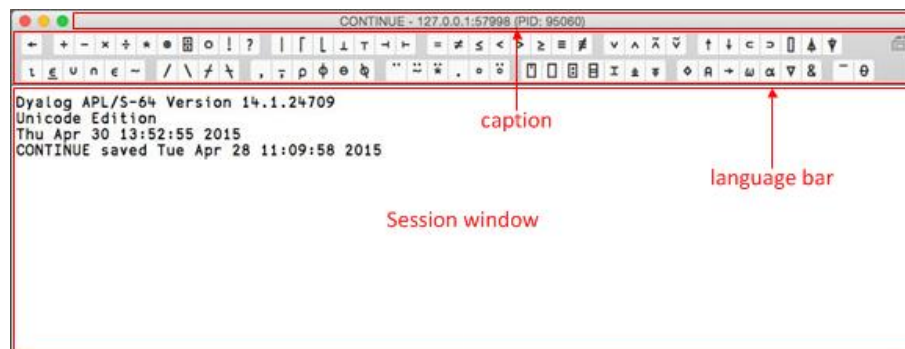


Figure 3. The Dyalog Session user interface (Mac OS)

5.3.1 Caption

The caption at the top of the **Session** Window details the ID of the workspace and the host, port and interpreter process numbers.


The caption can be customised (see section 9.2.5).



5.3.2 Language Bar

The language bar is located at the top of the **Session** window, beneath the caption. It contains buttons for each of the glyphs used as primitives in Dyalog.

When the cursor is positioned over one of the glyphs, information for that glyph is displayed. This includes the name of the glyph, the keyboard shortcut to enter it, its monadic/dyadic name and examples of its syntax, arguments and result.

Clicking on one of the glyphs copies that glyph into the active **Session/Edit** window at the position of the input cursor (the same as typing it directly into the **Session/Edit** window). If the an Edit window is the active window, it must be docked rather than floating for this to work (see section 7.3).

On the right hand side of the language bar is the  icon:

- Positioning the cursor over the  icon displays a tooltip showing selected keyboard shortcuts for command codes – for more information on these, see section 7.1.
- Clicking on the  icon displays the **Preferences** dialog box (the same as selecting the **Edit > Preferences** menu option) – for more information on the **Preferences** dialog box, see section 9.2.

Display of the language bar can be toggled with the **View > Show Language Bar** menu option or by entering the *Toggle Language Bar* command (<LBR>).

5.3.3 Session Window

The primary purpose of the **Session** window is to provide a scrolling area within which a user can enter APL expressions and view results. See section 6.1 for more information on the **Session** window.

You can move, resize, maximise and minimise the **Session** window using the standard facilities provided by your operating system.

6 Input Windows

Instead of just a single **Session** window, the Dyalog Development Environment can comprise multiple windows:

- **Session** window – created when a Dyalog Session is started through the RIDE and always present while the Session is live. There is only one **Session** window.
- **Edit** windows – created and destroyed dynamically as required. There can be multiple **Edit** windows (one for each APL object).
- **Trace** window – created and destroyed dynamically as required. There is only one **Trace** window.

Edit and **Trace** windows can be docked or floating (see section 7.3).

When multiple windows are open, the window that has the focus is referred to as the *active* window.

6.1 Session Window

The **Session** window contains:

- the *input line* – the last line entered in the **Session** window; this is (usually) the line into which you type an expression to be evaluated.
- the *Session log* – a history of previously-entered expressions and the results they produced.

If a log file is being used, then the Session log is loaded into memory when a Dyalog Session is started through the RIDE. When the Dyalog Session is closed, the Session log is written to the log file, replacing its previous contents.


6.2 Edit Window

The **Edit** window is used to define new objects as well as view/amend existing objects.

An **Edit** window can be opened from the **Session** window in any of the following ways:

- Enter `)ED <object name>`
- Enter `□ED '<object name>'`
- Enter `<object name> <ED>`
(for an explanation of the `<ED>` syntax, see section 7.1)
- Double-click on/after `<object name>`

If the object name does not already exist, then it is assumed to be of type function/operator. Different types can be explicitly specified using the `)ED` or `□ED` options – see `)ED` or `□ED` in the *Dyalog Language Reference Guide* for information.

An **Edit** window can be opened from the **Trace** window by entering the *Edit* command (**<ED>**), double-clicking the cursor or clicking the  button in the toolbar (see section 6.3.1). The position of the cursor when this is done determines the name of the object that the **Edit** window if for:

- If the cursor is on or immediately after `<object name>`, then the **Edit** window opens on that name.
- If the cursor is anywhere else, then the **Edit** window opens for the most recently-referenced function on the stack. This is a *naked edit*.

An **Edit** window can be opened from another **Edit** window in any of the following ways:

- Move the cursor over/after `<object name>` and enter the *Edit* command (**<ED>**)
- Double-click on/after `<object name>`

By default, the **Edit** window is docked to the right of the **Session** Window.





6.2.1 Toolbar



The toolbar displayed at the top of the **Edit** window is shown in figure 4; the icons on this toolbar are detailed in table 6.



Figure 4. The **Edit** Window's toolbar

Table 6. Icons on the **Edit** window's toolbar

Icon	Action	Description
	Toggle line numbers	Turns the display of line numbers on/off
	Comment selected text	Add a comment symbol (A) to the beginning of the line in which the cursor is positioned. If text has been selected, then a comment symbol is added at the start of the selection and at the start of each subsequent line of text within the selection. Same as the <i>Comment Out</i> command (<AO>).
	Uncomment selected text	Removes the comment symbol (A) at the beginning of the line in which the cursor is positioned. If text has been selected, then comment symbols are removed from the start of each selected line of text. Comment symbols that are not at the start of a line of text are not unremoved unless only the comment symbol and subsequent text on that line are selected. Same as the <i>Uncomment</i> command (<DO>).
	<i>Search and Replace fields</i>	See Section 6.2.2; the next three buttons relate to the <i>Search and Replace fields</i>
	Search for next match	Positions the cursor at the next occurrence of the <i>Search</i> text

	Search for previous match	Positions the cursor at the previous occurrence of the <i>Search</i> text
	Match case	Specifies whether the search is case-sensitive


6.2.2 Search and Replace

The *Search* and *Replace* fields on the **Edit** window's toolbar can be used to locate every occurrence of a specified string (this can include APL glyphs) within the code in the active **Edit** window; optionally, a replacement string can be applied on an individual basis.

To search for a string:

1. Enter the string to search for in the *Search* field in one of the following ways:
 - select the string and use the *Search* command (<SC>); the selected string is copied to the *Search* field.
 - enter the string directly in the *Search* field – use the *Search* command (<SC>) to move the cursor here.

All occurrences of the specified string are highlighted in the **Edit** window. If the content of the **Edit** window is sufficiently long for there to be a vertical scroll bar, then the locations of occurrences of the specified string within the entire content are identified by yellow marks overlaid on the scroll bar.


2. Press the **Enter** key to select the first occurrence of the search string after the last position of the cursor.
3. Press either the **Enter** key or the **Search for next match** button  to advance the selection to the next occurrence of the search string.
4. Repeat step 3 as required (the search is cyclic).
5. Use the *Escape* command (<EP>) or the **Esc** key to exit the search functionality.

To replace a string:

1. Enter the string to be replaced in the *Search* field in one of the following ways:
 - select the string and use the *Search* command (<SC>); the selected string is copied to the *Search* field.
 - enter the string directly in the *Search* field.

All occurrences of the specified string are highlighted in the **Edit** window. If the content of the **Edit** window is sufficiently long for there to be a vertical scroll bar, then the locations of occurrences of the specified string within the entire content are identified by yellow marks overlaid on the scroll bar.

2. Enter the replacement string directly in the *Replace* field.
3. Press the **Enter** key to select the first occurrence of the search string after the last position of the cursor.

4. Do one of the following:
 - Press the **Enter** key to replace the selected occurrence of the search string and to advance the selection to the next occurrence of the search string.
 - Press the **Search for next match** button  to leave the selected occurrence of the search string unaltered and advance the selection to the next occurrence of the search string.
5. Repeat step 0 until the required changes have been made (the search is cyclic).
6. Use the *Escape* command (<EP>) or the **Esc** key to exit the search and replace functionality.

6.2.3 Exiting the Edit Window

To save changes and close the **Edit** window:

- click on the **X** icon
- use the *Escape* command (<EP>)
- press the **Esc** key

To close the **Edit** window without saving changes:

- use the *Quit* command (<QT>)

6.3 Trace Window



The **Trace** window is read-only.

The **Trace** window aids debugging by enabling you to step through your code line by line, display variables in **Edit** windows and watch them change as the execution progresses. Alternatively, you can use the **Session** window and **Edit** windows to experiment with and correct your code.

A **Trace** window can be opened from the **Session** window by entering `<expression> <TC>`. This is an *explicit trace* and lets you step through the execution of any non-primitive functions/operators in the expression.

If there is a suspended function on the stack, then a **Trace** window can be opened from the **Session** window by double-clicking on a blank line – this can be a blank input line or a blank line in the Session log. A **Trace** window for the most recently-referenced function on the stack is displayed; this is a *naked trace*.

By default, Dyalog is also configured to initiate an *automatic trace* whenever an error occurs, that is, the **Trace** window opens and becomes the active window and the line that caused the execution to suspend is selected. This is controlled by the `TRACE_ON_ERROR` environment variable (for information on environment variables, see the *Dyalog for <operating system> Installation and Configuration Guide* specific to the operating system that you are using).

By default, the **Trace** window is docked beneath the **Session** and **Edit** windows. Other than setting/removing breakpoints (see section 7.7.1), **Trace** windows are read-only.

6.3.1 Toolbar

The toolbar displayed at the top of the **Trace** window is shown in figure 5; the icons on this toolbar are detailed in table 7.



Figure 5. The **Trace** Window's toolbar

Table 7. Icons on the **Trace** window's toolbar

Icon	Action	Description
	Execute line	Executes the current line and advances to the next line
	Trace into expression	Traces execution of the current line and advances to the next line. If the current line calls a user-defined function then this is also traced.
	Go back one line	Changes the currently-selected line to be the previous line in the code
	Skip current line	Changes the currently-selected line to be the next line in the code
	Stop on next line of calling function	Continues execution of the code in the Trace window from the current line to completion of the current function or operator, leaving the Trace window open. If successful, the code is cut from the stack (its Trace window is closed) and the selection advances to the next line of the calling function (if there is one).
	Continue execution of this thread	Closes the Trace window and resumes execution of the current application thread from the current line.
	Continue execution of all threads	Closes the Trace window and resumes execution of all suspended threads.
	Edit name	Converts the Trace window into an Edit window as long as the cursor is on a blank line or in an empty space. However, if the cursor is on or immediately after an object name that is not the name of the suspended function, then an Edit window for that object name is opened.
	Interrupt	Interrupts execution with a weak interrupt
	Clear breakpoints for this object	Clears all breakpoints (resets <input type="checkbox"/> STOP on every function) for the object
	Toggle line numbers	Turns the display of line numbers on/off
	<i>Search field</i>	See Section 6.3.2; the next three buttons relate to the <i>Search field</i>
	Search for next match	Locates the next occurrence of the <i>Search</i> text
	Search for previous match	Locates the previous occurrence of the <i>Search</i> text

aA	Match case	Specifies whether the search is case-sensitive
X	Quit this function	Closes the Trace window and cuts the code that it contained it from the stack.


6.3.2 Search

The *Search* field on the **Trace** window's toolbar can be used to locate every occurrence of a specified string (this can include APL glyphs) within the code in the **Trace** window.

To search for a string:

1. Enter the string to search for in the *Search* field in one of the following ways:
 - select the string and use the *Search* command (<SC>); the selected string is copied to the *Search* field.
 - enter the string directly in the *Search* field – use the *Search* command (<SC>) to move the cursor here.

All occurrences of the specified string are highlighted in the **Trace** window. If the content of the **Trace** window is sufficiently long for there to be a vertical scroll bar, then the locations of occurrences of the specified string within the entire content are identified by yellow marks overlaid on the scroll bar.

2. Press the **Enter** key to select the first occurrence of the search string after the last position of the cursor.
3. Press either the **Enter** key or the **Search for next match** button  to advance the selection to the next occurrence of the search string.
4. Repeat step 3 as required (the search is cyclic).
5. Use the *Escape* command (<EP>) or the **Esc** key to exit the search functionality.

6.3.3 Exiting the Trace Window

To close the **Trace** window:

- click on the **X** icon
- use the *Escape* command (<EP>)
- use the *Quit* command (<QT>)
- press the **Esc** key


When the **Trace** window is closed, the function within that **Trace** window is removed from the stack. It is possible to close all **Trace/Edit** windows without clearing the stack using 2023I – see the *Dyalog Language Reference Guide*.

7 Working in a Dyalog Session

The main purpose of a development environment is to enable a user to enter and execute expressions; this chapter describes how this can be achieved when running a Dyalog Session through the RIDE and explains the functionality that is provided to simplify the process.

7.1 Keyboard Shortcuts and Command Codes

Keyboard shortcuts are keystrokes that execute an action rather than produce a symbol. The RIDE supports numerous keyboard shortcuts, each of which is identified by a *command code* and mapped to a keystroke combination; for example, the action to open the **Trace** window is identified by the code **TC** (described in the documentation as **<TC>**). For a complete list of the command codes that can be used in a Dyalog Session running through the RIDE and the keyboard shortcuts for those command codes, see Appendix B.

Positioning the cursor over the  icon in the language bar displays a tooltip showing selected keyboard shortcuts for command codes.

Keyboard shortcuts can be customised (see section 9.2.2).

7.2 Navigating the Windows

When multiple windows are open, the window that has the focus is referred to as the *active* window. A window can be made the active window by clicking within it.

The **Session**, **Edit** and **Trace** windows form a closed loop for the purpose of navigation:

- to make the next window in this loop the active window, enter the *Tab Window* command (**<TB>**)
- to make the previous window in this loop the active window, enter the *Back Tab Window* command (**<BT>**)

An active **Edit/Trace** window can be closed after changes have been made to its content:

- to save any changes in the content of the active window before closing it, enter the *Escape* command (**<EP>**) or press the **Esc** key
- to discard any changes in the content of the active window before closing it, enter the *Quit* command (**<QT>**)

7.3 Docking and Floating Windows

By default, the **Trace** window and **Edit** windows are docked within the Session. If the menu option **View > Float New Editors** is checked, then any windows created subsequently are not docked but floating.

Changing the setting of **View > Float New Editors** does not affect windows that already exist.

Docked windows can be selected (by clicking within them) and resized (by moving the splitter bar). Floating windows can be selected, moved, resized, maximised and minimised using the standard facilities that the operating system provides.



The language bar (see section 5.3.2) is displayed in the **Session** window; it can only be used in the **Session** window and docked **Edit** windows (not in floating **Edit** windows).

7.4 Entering APL Characters

APL glyphs can be entered in a Dyalog Session running through the RIDE by:

- typing the glyph in the **Session** window or **Edit** window using the appropriate key combination (see section 5.2).
- clicking the appropriate glyph on the language bar (see section 5.3.2) – this inserts that glyph into the active **Session/Edit** window at the position of the cursor.

When typing a glyph directly rather than using the language bar, if you pause after entering the prefix key then the autocomplete functionality (see section 7.5.2) displays a list of all the glyphs that can be produced. If you enter the prefix key a second time then a list of all the glyphs that can be produced is again displayed but this time with the names (formal and informal) that are used for each glyph.

For example:

```
⊛ A default prefix key
```

The autocomplete functionality list includes the following for the ⊛ glyph:

```
⊛ \* ``logarithm
⊛ \* ``naturallogarithm
⊛ \* ``circlestar
⊛ \* ``starcircle
⊛ \* ``splat
```

This means that you can enter the ⊛ glyph by selecting (or directly typing) any of the following:

```
\*
``logarithm
``naturallogarithm
``circlestar
``starcircle
``splat
```

As you enter a name, the autocomplete functionality restricts the list of options to those that match the entered name.

For example, entering:

```
``ci
```

restricts the list to:

```
⊛ `* ``circlestar
o `o ``circular
φ `% ``circlestile
e `& ``circlebar
⊘ `^ ``circlebackslash
ö `O ``circlediaeresis
```

7.5 Entering Expressions

The RIDE provides several mechanisms that assist with accuracy and provide clarity when entering expressions in a Dyalog Session.

7.5.1 Paired Enclosures

*Applicable in the **Session** window and the **Edit** window.*

Enclosures in the RIDE include:

- parentheses ()
- braces { }
- brackets []

Angle brackets < > are not enclosures.

When an opening enclosure character is entered, the RIDE automatically includes its closing pair. This reduces the risk of an invalid expression being entered due to unbalanced enclosures. This feature can be disabled in the **Code** tab of the **Preferences** dialog box (see section 9.2.3).

7.5.2 Autocomplete

*Applicable in the **Session** window and the **Edit** window.*

The RIDE includes autocomplete functionality for names to reduce the likelihood of errors when including them in an expression (and to save the user having to enter complete names or remember cases for case-sensitive names).

As a name is entered, the RIDE displays a pop-up window of suggestions based on the characters already entered and the context in which the name is being used.

For example, if you enter a `[` character, the pop-up list of suggestions includes all the system names (for example, system functions and system variables). Entering further characters filters the list so that only those system functions and variables that start with the exact string entered are included.

When you start to enter a name in the **Session** window, the pop-up list of suggestions includes all the namespaces, variables, functions and operators that are defined in the current namespace. When you start to enter a name in the **Edit** window, the pop-up list of suggestions also includes all names that are localised in the function header.

To select a name from the pop-up list of suggestions, do one of the following:

- click the mouse on the name in the pop-up list
- use the right arrow key to select the top name in the pop-up list
- use the up and down arrow keys to navigate through the suggestions and the right arrow key or the **ENTER** key to enter the currently-highlighted name

The selected name is then completed in the appropriate window.

This feature can be disabled in the **Code** tab of the **Preferences** dialog box (see section 9.2.3).

7.5.3 Context-Sensitive Help

*Applicable in the **Session** window, **Edit** window and **Trace** window*

With the cursor on or immediately after any system command, system name, control structure keyword or primitive glyph, enter the *Help* command (**<HLP>**). The documentation for that system command, system name, control structure keyword or primitive glyph will be displayed.

7.5.4 Syntax Colouring

*Applicable in the **Edit** window and **Trace** window*

Syntax colouring assigns different colours to various components, making them easily identifiable. The syntax colouring convention used is detailed in table 8.

Table 8. Syntax colouring convention

Colour	Syntax
black	global names
grey	names namespaces numbers tradfn syntax (header line and final ▽ in scripted syntax)
maroon	control structure keywords
red	errors (including unmatched parentheses, quotes and braces)
teal	strings comments
navy	primitive functions zilde
blue	idioms (this takes priority over any other syntax colouring) operators parentheses/braces/brackets dfn syntax (specifically, { } α ω ▽ and :) assignment (←), diamond (◊) and semi-colon (;)
purple	system names

Syntax colouring can be customised (see section 9.2.4).

7.6 Executing Expressions

7.6.1 Executing a New Expression

*Applicable in the **Session** window*

After entering a new expression in the input line, that expression is executed by pressing the **Enter** key or with the *Enter* command (**<ER>**). Following execution, the expression (and any displayed results) become part of the Session log.

7.6.2 Re-executing a Previous Expression

*Applicable in the **Session** window*

Instead of entering a new expression in the input line, you can move back through the Session log and re-execute a previously-entered expression.

To re-execute a previously-entered expression:

1. Locate the expression to re-execute in one of the following ways:
 - Scroll back through the Session log.
 - Use the *Backward* command (**<BK>**) and the *Forward* command (**<FD>**) to cycle backwards and forwards through the input history, successively copying previously-entered expressions into the input line.
2. Position the cursor anywhere within the expression that you want to re-execute and press the **Enter** key or use the *Enter* command (**<ER>**).

If required, a previously-entered expression can be amended prior to execution. In this situation, when the amended expression is executed it is copied to the input line; the original expression in the Session log is not changed. If you start to edit a previous expression and then decide not to, use the *Quit* command (**<QT>**) to return the previous expression to its unaltered state.

7.6.3 Re-executing Multiple Previous Expressions

*Applicable in the **Session** window*

Multiple expressions can be re-executed together irrespective of whether they were originally executed sequentially (certain system commands cause re-execution to stop once they have been completed, for example, **)LOAD** and **)CLEAR**).

To re-execute multiple previously-entered expressions:

1. Locate the first expression to re-execute in one of the following ways:
 - Scroll back through the Session log.
 - Use the *Backward* command (**<BK>**) and the *Forward* command (**<FD>**) to cycle backwards and forwards through the input history.
2. Change the expression in some way. The change doesn't have to impact the purpose of the expression; it could be an additional space character.
3. Scroll through the Session log to locate the next expression to re-execute and change it in some way. Repeat until all the required expressions have been changed.
4. Press the **Enter** key or enter the *Enter* command (**<ER>**)

The amended expressions are copied to the input line and executed in the order in which they appear in the Session log; the modified expressions in the Session log are restored to their original content.

To re-execute contiguous previously-entered expressions:

1. Position the cursor at the start of the first expression to re-execute.
2. Press and hold the mouse button (left-click)+ and drag the cursor to the end of the last expression to re-execute.
3. Copy the selected lines to the clipboard using the *Copy* command (<CP>) or the *Cut* command (<CT>) or the **Copy/Cut** options in the **Edit** menu.
4. Position the cursor in the input line and paste the content of the clipboard back into the Session using the *Paste* command (<PT>), the **Paste** option in the **Edit** menu or the **Paste** option in the context menu.
5. Press the **Enter** key or enter the *Enter* command (<ER>).

This technique can also be used to move lines from the **Edit** window into the **Session** window and execute them.

7.7 Suspending Execution

To assist with investigations into the behaviour of a set of statements (debugging), the system can be instructed to suspend execution just before a particular statement. This is done by setting a breakpoint – see section 7.7.1.

It is sometimes necessary to suspend the execution of a function, for example, if an endless loop has been inadvertently created or a response is taking an unacceptably long time. This is done using an interrupt – see section 7.7.2.

Suspended functions can be viewed through the stack; the most recently-referenced function is at the top of the stack. The content of the stack can be queried with the `)SI` system command; this generates a list of all suspended and pendent (that is, awaiting the return of a called function) functions, where suspended functions are indicated by a `*`. For more information on the stack and the state indicator, see the *Dyalog Programmer's Guide*.

7.7.1 Breakpoints

*Applicable in the **Edit** window and the **Trace** window.*

When a function that includes a breakpoint is run, its execution is suspended immediately before executing the line on which the breakpoint is set and the **Trace** window is automatically opened (assuming that automatic trace is enabled – see section 6.3).

Breakpoints are defined by dyadic `□STOP` and can be toggled on and off in an **Edit** or **Trace** window by left-clicking on the far left of the line before which the break point is to be applied or by placing the cursor anywhere in the line before which the break point is to be applied and entering the *Toggle Breakpoint* command (<BP>). Note that:

- Breakpoints set or cleared in an **Edit** window are not established until the function is fixed.
- Breakpoints set or cleared in a **Trace** window are established immediately.



When a breakpoint is reached during code execution, event 1001 is generated; this can be trapped. For more information, see `□TRAP` in the *Dyalog Language Reference Guide*.

7.7.2 Interrupts

A Dyalog Session running through the RIDE responds to both strong and weak interrupts.

Entering a *strong interrupt* suspends execution as soon as possible (generally after completing execution of the primitive currently being processed). A strong interrupt is issued by selecting **Actions > Strong Interrupt** in the menu options or by entering the *Strong Interrupt* command (<SI>).

Entering a *weak interrupt* suspends execution at the start of the next line (generally after completing execution of the statement currently being processed). A weak interrupt is issued by selecting **Actions > Weak Interrupt** in the menu options or by entering the *Weak Interrupt* command (<WI>).



When a strong or weak interrupt is issued during code execution, event 1003 or 1002 (respectively) is generated; these can be trapped. For more information, see `TRAP` in the *Dyalog Language Reference Guide*.

7.8 Terminating a Dyalog Session Running Through the RIDE

The Dyalog Session can be terminated (without having to close any open windows first) in any of the following ways:

- From the menu options:
 - Linux: Select **File > Quit**
 - Mac OS : **Dyalog > Quit Dyalog**
 - Microsoft Windows: Select **File > Quit**
- Enter:
 - Linux: **Ctrl + Q**
 - Mac OS : **⌘ + Q**
 - Microsoft Windows: **Ctrl + Q**
(only if the Dyalog Unicode IME is not enabled – see section 9.2.1)

In addition, when the **Session** window is the active window, the Dyalog Session can be terminated cleanly in any of the following ways:

- Enter `)OFF`
- Enter `␣OFF`
- Enter `<QIT>`
- Click the close button
- Mac OS: enter **⌘ + W**

8 RIDE-Specific Language Features

When running a Dyalog Session through the RIDE, the majority of language features remain unaltered. However, there are a few additional features and some existing functionality that is meaningless when a Session is running through the RIDE.

8.1 I-Beams

I-Beam is a monadic operator that provides a range of system-related services.

Syntax: $R \leftarrow \{X\} (A \text{I}) Y$



Any service provided using an I-Beam should be considered as experimental and subject to change – without notice – from one release to the next. Any use of I-Beams in applications should, therefore, be carefully isolated in cover-functions that can be adjusted if necessary.

There are three I-Beams that are only relevant to the RIDE.

8.1.1 3500I : Send Text to RIDE-embedded Browser

Syntax: $R \leftarrow \{X\} (3500I) Y$

Optionally, X is a simple character vector or scalar, the contents of which are used as the caption in the RIDE client that contains the embedded browser. If omitted, then the caption defaults to "3500I".

Y is a simple character vector the contents of which are displayed in the embedded browser tab.

R identifies whether the write to the RIDE was successful. Possible values are:

- 0 : the write to the RIDE client was successful
- -1 : the RIDE client is not enabled
- any other value : contact support@dyalog.com

8.1.2 3501I : Connected to the RIDE?

Syntax: $R \leftarrow \{X\} (3501I) Y$

X and Y can be any value (ignored).

R identifies whether the Dyalog Session is running through the RIDE. Possible values are:

- 0 : the Session is not running through the RIDE
- 1 : the Session is running through the RIDE

8.1.3 3502I : Enable RIDE in Run-time Interpreter



This I-Beam is only relevant for run-time interpreters on the Microsoft Windows operating system.

Syntax: 3502Iθ

By default, the RIDE is not enabled on run-time executables. For security reasons, enabling the RIDE is a two-step process rather than using (for example) a single environment variable. To enable the RIDE, two steps must be taken:

1. Set the RIDE_INIT configuration parameter (see section 8.2.1) on the machine on which the run-time interpreter is running to an appropriate value.
2. Execute 3502Iθ in your application code

The run-time interpreter can then attempt to connect to a RIDE client.



Enabling the RIDE to access applications that use the run-time interpreter means that the APL code of those applications can be accessed. The I-Beam mechanism described above means that the APL code itself must grant the right for a RIDE client to connect to the run-time interpreter. Although Dyalog Ltd might change the details of this mechanism, the APL code will always need to grant connection rights. In particular, no mechanism that is only dependent on configuration parameters will be implemented.

8.2 Configuration Parameters

There is one configuration parameter that is relevant to the RIDE:

- RIDE_INIT

8.2.1 RIDE_INIT

How the interpreter should behave with respect to the RIDE protocol. Setting this configuration parameter on the machine that hosts the interpreter enables the interpreter-RIDE connection.

The format of the value is <setting> : <address> : <port>

where:

- <setting> is the action the interpreter should take. Valid values are:
 - *serve* – listen for incoming connections
 - *connect* – connect to the specified RIDE and end the session if this fails
 - *poll* – try to connect to the specified RIDE at regular intervals and reconnect if the connection is lost
- <address> is the machine on which to listen for a connection (if <setting> is *serve*) or connect to (if <setting> is *connect/poll*). Valid values are:
 - the name of the machine
 - the IPv4 address of the machine
 - the IPv6 of the machine

- o *<empty>* – if *<setting>* is *serve* then the interpreter listens to everything on every network interface, if *<setting>* is *connect/poll* then the interpreter only listens for local connections (127.0.0.1).
- *port* is the TCP port to listen on

The RIDE_INIT configuration parameter is set automatically when launching a new Dyalog Session from the RIDE (see section 4.1.2).

8.3 Unsupported Language Elements

When running a Dyalog Session through the RIDE, a few of the features that are available with non-RIDE Sessions do not function as might be expected.

8.3.1 Function Key Configuration

Character strings (including command keys) can be associated with programmable function keys using the `PFKEY` system function. When running a Dyalog Session through the RIDE, `PFKEY` can be used to define/display the keystrokes for a designated function key; however, that function key does not acquire the defined set of keystrokes, rendering `PFKEY` of no real use.

This restriction might be removed in a future release of the RIDE.

8.3.2 Underscored Characters

Underscored characters can be entered into the **Session** window and **Edit** windows using the ```_<letter>` method, for example, enter ```_f` to produce F.

8.3.2.1 Underscored Characters in Window Captions

The RIDE is restricted by the operating system when it comes to displaying underscored characters in window titles (captions). This restriction means that:

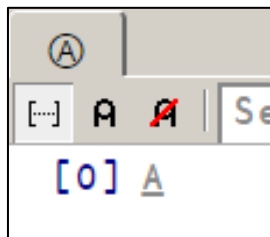
- if the APL385 font is installed, underscored characters are displayed as circled characters
- if the APL385 font is not installed underscored characters are displayed as `⎕`

For example:

Open an **Edit** window for an object that has an underscored name:

```
)ed A
```

The **Edit** window that is opened displays the name correctly, but its title (caption) is displayed incorrectly, as shown (assuming window is docked):



8.3.2.2 Underscored Characters in the Session

If the RIDE is connected to a Unicode edition of Dyalog, then underscored characters are displayed correctly in the **Session** window, **Edit** windows and **Trace** windows. If the RIDE is connected to a classic edition of Dyalog, then the following command must be run in every APL Session to enable the underscored alphabet to be displayed correctly:

```
⊞IO←0
⊞AVU[97+ι26]←9398+ι26
2 ⊞NQ '.' 'SetUnicodeTable' ⊞AVU
```

8.3.3 Operating System Terminal/Command Window Interaction

Features that rely on interaction with an operating system terminal/command window (that is, ⊞SR and)SH or)CMD with no argument) cannot work in a Dyalog Session that is running through the RIDE. Instead of behaving as documented in the *Dyalog Language Reference Guide*, their behaviour depends on the way in which the interpreter and the RIDE combined to start a Dyalog Session:

- If the interpreter was started by the RIDE (see section 4.1.2), then:
 - ⊞SR generates a trappable error
 -)SH or)CMD with no argument produces a "Feature disabled in this environment" message.
- If the interpreter and the RIDE were started independently and then connected to each other (see sections 4.1.1 and 4.1.3), then the use of these features will appear to hang the Dyalog Session that is running through the RIDE. However, the Dyalog Session can be recovered by locating the operating system terminal/command window and using it to complete the operation. If there is no operating system terminal/command window, then the Dyalog Session is irrecoverable.

9 Customising Your Session

The appearance and behaviour of a Dyalog Session running through the RIDE can be customised to meet personal preferences and corporate guidelines. Configuration can be performed:

- through the **View** menu – see section 9.1
- through the **Preferences** dialog box – see section 9.2
- using environment variables – see section 9.3

Customisations performed using any of these methods persist between Sessions (they also persist when the installed version of Dyalog is upgraded).



To remove all customisations, reset all RIDE-specific settings and return to the initial default settings, rename/delete the following directory:

- Microsoft Windows: `%LOCALAPPDATA%\ride<version>`
 - Linux: `$HOME/.config/ride<version>`
 - Mac OS: `$HOME/Library/Application Support/ride<version>` (hidden directory – access from the command line)
-

9.1 View Menu

The **View** menu (see section 5.1.3) includes options that enable the appearance of the Dyalog Session running through the RIDE to be changed. Select these options to:


- show/hide the language bar (see section 5.3.2)
- float/dock new **Edit** and **Trace** windows (see section 7.3)
- enable/disable automatic line wrapping in **Session** window
- always display floating **Edit** windows on top of the **Session** window
- change the font size in all windows in the Session
- change the overall look of the windows by selecting a pre-defined theme

9.2 Preferences Dialog Box

The **Preferences** dialog box can be used to customise:

- the default keyboard key mappings for APL glyphs (see section 9.2.1)
- the keyboard shortcuts for command codes (see section 9.2.2)
- the automatic formatting of text in an **Edit** window (see section 9.2.3)
- the syntax colouring and background colour (see section 9.2.4)
- the caption of the **Session** window (see section 9.2.5)
- the menu options presented by the menu bar (see section 9.2.6)

The **Preferences** dialog box can be opened in any of the following ways:

- selecting the **Edit > Preferences** menu option
- clicking the  icon on the right hand side of the language bar
- entering the *Show Preferences* command (`<PRF>`)

9.2.1 Layout Tab

Allows customisation of the default keyboard key mappings for APL glyphs (see section 5.2). This is only relevant if a locale-specific keyboard has not been installed.



To replace the keyboard with a locale-specific keyboard in the Session, or to enter Dyalog glyphs in other applications (for example, email), see <http://www.dyalog.com/apl-font-keyboard.htm>.



If you have the Dyalog Unicode IME installed, then the RIDE activates it at start-up when the **Select Dyalog IME as the active keyboard** checkbox is selected (selected by default).

If Dyalog is not installed on the machine that the RIDE is running on, then the Dyalog Unicode IME can be downloaded and installed from <http://www.dyalog.com/apl-font-keyboard.htm>.



Most Linux distributions released after mid-2012 support Dyalog glyphs by default, for example, openSUSE 12.2, Ubuntu 12.10 and Fedora 17. For more information, see the *Dyalog for UNIX Installation and Configuration Guide*.

The default keyboard that is installed with the RIDE for use in a Session is shown in Appendix A (this shows the UK keyboard – other locales are available through a drop-down list).

To customise the default keyboard's Prefix key

1. Open the **Layout** tab and select the appropriate keyboard from the drop-down list of options.
2. In the **Prefix** key field, enter the new prefix key (by default this is `).



In locales in which ` is a *dead key*, \$ is a viable alternative.

Be careful when selecting a new prefix key – although there are no restrictions, choosing certain keys (for example, alphanumeric characters) would restrict the information that could be entered in a Session.

3. Click **OK** to save your changes or **Cancel** to close the **Preferences** dialog box without saving your changes.

To customise the default keyboard's key mappings

1. Open the **Layout** tab and select the appropriate keyboard from the drop-down list of options.
2. In the image of a keyboard, click on the glyph to be replaced.
3. In the language bar, click on the glyph to replace the selected glyph with.
4. Repeat steps 2 and 3 until the key mappings are as required.
5. Click **OK** to save your changes and close the **Preferences** dialog box, **Apply** to save your changes without closing the **Preferences** dialog box or **Cancel** to close the **Preferences** dialog box without saving your changes.

Your selection of keyboard key mappings is unrestricted – you can choose to map the same glyph to multiple keys or have glyphs that are not represented on the keyboard at all. For example, when dealing with dfns on a Danish keyboard, it might be convenient to map { and } to a simpler key combination.


9.2.2 Shortcuts Tab

Allows customisation of the keyboard shortcuts for command codes (see section 7.1 and Appendix B). Multiple shortcuts can be defined for any command code, but each shortcut must be unique.

To change the keyboard shortcut for a command code

1. Open the **Shortcuts** tab.
2. Locate the command code that you want to define a new keyboard shortcut for. This can be done by scrolling through the list of possible command codes or by entering a string in the *Search* field. If a string is entered in the search field then a dynamic search of both the command codes and descriptions is performed.
3. Optionally, delete any existing shortcuts for the command by clicking the cross to the right of the shortcut (this will turn red when moused over).
4. Optionally, add a new shortcut. To do this:
 - a. Click the plus symbol to the right of any existing shortcuts (this will turn green when moused over). The **New Shortcut** dialog box will be displayed
 - b. In the **New Shortcut** dialog box, enter the keystrokes to map to this action. The **New Shortcut** dialog box closes when the keystrokes have been entered and the new shortcut is displayed on the **Shortcuts** tab.

If the keystrokes that you enter are already used for a different command code, then both occurrences will be highlighted and you should remove any duplicate entries (an error message will be displayed if you attempt to apply/save settings that contain duplicate entries).
5. Click **OK** to save your changes and close the **Preferences** dialog box, **Apply** to save your changes without closing the **Preferences** dialog box or **Cancel** to close the **Preferences** dialog box without saving your changes.

The tooltip showing selected keyboard shortcuts for command codes (obtained by positioning the cursor over the  icon in the language bar) is updated to reflect the customisation if relevant.

9.2.3 Code Tab

Allows customisation of the automatic formatting/layout of text in an **Edit** window.

To change the way in which text is automatically formatted in the Edit window

1. Open the **Code** tab.
2. Select the appropriate check boxes and set the variables as required:
 - *Auto-indent <number> spaces*
Select this and define the number of spaces by which each level of nested code should be indented relative to the previous level of indentation in the **Edit** window. If not selected, code in the **Edit** window will be left justified.
 - *in methods: <number> spaces*
Select this and define the number of spaces by which the contents of tradfns should be indented relative to the start/end ▽ glyphs.

When changes to auto-indentation are applied, the indentation of existing code in an open **Edit** window does not change unless you enter the *Reformat* command (<RD>) when the **Edit** window is the active window. However, any new code entered in the **Edit** window follows the new rules.

- *Indent lines that contain only a comment*
Select this to apply the appropriate indentation to lines that start with the ¶ glyph. If this is not selected, then lines that start with the ¶ glyph remain as positioned by the user.
- *Indent content when an editor is opened*
Select this to apply the indentation rules to the contents of an **Edit** window when it is opened.
- *Highlight matching brackets: () [] {}*
Select this to automatically highlight the matching start and end enclosures when positioning the cursor before or after one of them (with contiguous brackets, the bracket immediately before the cursor has its other enclosure highlighted).
- *Auto-close brackets*
Select this to automatically add the paired enclosure when an opening enclosure is entered (see section 7.5.1).
- *Auto-close blocks: :If :For ... with <select>*
Select this to automatically insert matching end statements when an opening control structure statement is entered.
- *Autocompletion after <time> ms*
Select this and specify a time interval after which the RIDE will display a pop-up window of suggestions based on the characters already entered and the context in which a name is being used (see section 7.5.2).
- *Code folding (outlining)*
Select this to enable code folding/outlining of control structures (including :Section structures) and functions. When changes to outlining are applied, existing code in an open **Edit** window is automatically updated to reflect the new rules.

3. Click **OK** to save your changes and close the **Preferences** dialog box, **Apply** to save your changes without closing the **Preferences** dialog box or **Cancel** to close the **Preferences** dialog box without saving your changes.



Settings that impact the automatic reformatting of code can cause changes to whitespace – this can be interpreted as changes to the source code. This means that:

- opening a scripted object in the **Edit** window can cause the source of that object to change (when closing an **Edit** window, you might be prompted to save a function even though you have not made any changes to it).
 - viewing an object can change its file timestamp; source code management systems can subsequently report changes due to the changed file timestamp.
 - source code changes resulting from reformatting will be evident in the results of system functions such as `⎕AT`, `⎕SRC`, `⎕CR`, `⎕VR` and `⎕NR`.
-

9.2.4 Colours Tab

Allows customisation of the syntax colouring (see section 7.5.4). Several schemes are provided, any of which can be used as they are or further customised.

To change the syntax colouring to a predefined scheme

1. Open the **Colours** tab.
2. In the **Scheme** field, select the syntax colouring scheme that you want to use. When a scheme is selected, the example shown is updated to use that colour scheme.
3. Click **OK** to save your changes and close the **Preferences** dialog box, **Apply** to save your changes without closing the **Preferences** dialog box or **Cancel** to close the **Preferences** dialog box without saving your changes.

To define a new syntax colouring scheme

1. Open the **Colours** tab.
2. In the **Scheme** field, select the syntax colouring scheme that is closest to the scheme that you want to define. When a scheme is selected, the example shown is updated to use that colour scheme.
3. Click **Clone**.
A copy is made of the selected scheme and additional options are displayed to allow customisation of the scheme's name and syntax colouring.
4. Define your colour scheme. To do this:
 - a. Select the element that you would like to change the display of from the drop-down list or by clicking in the example. The customisation options reflect the current settings for the selection made. Some elements have a limited set of options, for example, the *cursor* element has no bold/italic/underline option.
 - b. Select a foreground colour and background (highlighting) colour for that syntax as required. If a background colour is selected, then the slide bar can be used to set its transparency.

- c. Select the appropriate check boxes to make that syntax bold, italic or underlined as required.
 - d. Repeat as required.
5. Optionally, rename your colour scheme. To do this:
 - a. Click **Rename**.
The name in the Scheme field becomes editable.
 - b. Edit the name in the **Scheme** field.
 6. Click **OK** to save your changes and close the **Preferences** dialog box, **Apply** to save your changes without closing the **Preferences** dialog box or **Cancel** to close the **Preferences** dialog box without saving your changes.

9.2.5 Title Tab

Allows customisation of the caption at the top of the **Session** window (see section 5.3.1).

To change the Session window caption

1. Open the **Title** tab.
2. In the **Window title** field, enter the new name for the Session. Some variable options that can be included are listed beneath this field – clicking on these inserts them into the **Window title** field.
3. Click **OK** to save your changes and close the **Preferences** dialog box, **Apply** to save your changes without closing the **Preferences** dialog box or **Cancel** to close the **Preferences** dialog box without saving your changes.

The default value is {WSID}.

Setting this to {WSID} - {HOST} : {PORT} (PID: {PID}) gives a different result on every machine. For example:

/Users/nick/myws.dws - 127.0.0.1:49376 (PID: 293)

Changing this to {CHARS} {BITS} gives information that could be the same on multiple machines:

Unicode 64

9.2.6 Menu Tab

Allows customisation of the menu options in the menu bar (see section 5.1).



Great care should be taken when customising the menu options; although the ability to make changes is provided, it is not an activity that Dyalog Ltd supports.

If menu options are customised, then updates to menu items are ignored when updating the installed version of RIDE (this can be avoided by resetting the menu options before upgrading RIDE).

Top-level options (menu names in the menu bar) can be:

- created
- renamed
- reordered
- deleted

Options within each of the menus can be:

- moved to be under a different menu name in the menu bar
- reordered under the same menu name in the menu bar
- renamed
- deleted

Changes do not take effect until the next time a Dyalog Session is started through the RIDE.

9.3 Environment Variables

Some customisation can be performed using environment variables outside a Session. For details of other environment variables that can be set, and the syntax used to set them, see the *Dyalog for <operating system> Installation and Configuration Guide* specific to the operating system that you are using.

Appendix A Default Keyboard

The keyboard key mappings shown in figure 6 are the default mappings enabled when a Dyalog Session runs through the RIDE under a UK locale.

~	! I	" ' & #	3 <	4 ≤	5 =	% φ	^ q	2 e	* @	(~)	~ ^	+ +	Backspace
1	2	3	4	5	6	7	8	9	0	^	÷		
Tab	Q	W	E	R	T	Y	U	I	O	P	{ }	Enter	
	q	w	e	r	t	y	u	i	o	p	[]		
Caps Lock	A	S	D	F	G	H	J	K	L	:	~		
	a	s	d	f	g	h	j	k	l	;	#		
Shift		Z	X	C	V	B	N	M	<	>	? /	Shift	
	/	z	x	c	v	b	n	m	,	.	/		
Ctrl	Win	Alt							Alt	Gr	Win	Menu	Ctrl

Figure 6. The default Session keyboard key mappings (shown on a UK keyboard).

To access the glyphs in the lower right quadrant, press ` followed by the appropriate key.

To access the glyphs in the upper right quadrant, press ` followed by the **SHIFT** key with the appropriate key.

Appendix B Keyboard Shortcuts

The Dyalog keyboard shortcuts that are supported on the RIDE are listed in *Table 10*; those that can be configured in the **Shortcuts** tab of the **Preferences** dialog box (see section 9.2.2) are indicated with a * character. Keyboard shortcuts that are not currently supported on the RIDE are detailed in *Table 11*.

Table 10. Dyalog keyboard shortcuts supported on the RIDE

Keycode	Command	Default Keystrokes	Description
ABT *	About	Shift + F1	Display the About dialog box
AC *	Align Comments		Edit: Align comments to current column
AO *	Comment Out		Edit: Add comment symbol at start of each tagged or current line
BK *	Backward	Ctrl + Shift + Backspace	Session: Show previous line in input history Edit: Undo last change (where possible) Trace: Skip back one line
BP *	Toggle Breakpoint		Edit: Toggle a breakpoint on the current line Trace: Toggle a breakpoint on the current line
BT *	Back Tab Window	Ctrl + Shift + Tab	Move to previous window in loop
CNC *	Connect		Display the Connect to an interpreter dialog box
CP	Copy	Linux: Ctrl + C Mac: ⌘ + C Win: Ctrl + C	Session/Edit: Copy highlighted block of text to the clipboard
CT	Cut	Linux: Ctrl + X Mac: ⌘ + X Win: Ctrl + X	Session: Delete highlighted block of text and place it on the clipboard Edit: Delete highlighted block of text and place it on the clipboard
DB	Backspace	Backspace	Delete character to left of cursor
DC	Down Cursor	Down Arrow	Move cursor down one character

DI	Delete Item	Linux: Delete Mac: Fn + Backspace or Delete Win: Delete	Delete character to right of cursor
DK	Delete Block	Delete	Session: Delete highlighted block of text Edit: Delete highlighted block of text
DL	Down Limit	Linux: Ctrl + End Mac: ⌘ + Down Arrow or ⌘ + Fn + Right Arrow Win: Ctrl + End	Session: Move cursor to bottom right corner of Session log Edit: Move cursor to bottom right corner of content
DMK *	Toggle key display mode		Functionality that could be useful when presenting demonstrations. Enable you to display your keystrokes and load/run a demo file.
DMN *	Next line in demo		
DMP *	Previous line in demo		
DMR *	Load demo file		
DO *	Uncomment		Edit: Remove comment symbol that is first non-space character on each tagged or current line
DS	Down Screen	Linux: Page Down Mac: Fn + Down Arrow Win: Page Down	Move cursor down one screen
ED *	Edit	Shift + Enter	Session: Open an Edit window (if there is a suspended function on the stack, this opens an Edit window for that function – this is called <i>Naked Edit</i>)
EP *	Escape	Escape	Edit: Fix and Close Trace: Cut stack back to calling function; close all windows to match new stack status
ER *	Enter	Enter	Session: Execute current line/all modified lines Edit: Insert new line Trace: Execute current line

EXP *	Expand Selection	Shift + Alt + Up Arrow	Successive presses of <EXP> expand the highlighted selection Session: select the: <ul style="list-style-type: none"> • current line • entire Session log Edit: select the: <ul style="list-style-type: none"> • current token (number, name, string, and so on) • current line • entire contents of window
FD *	Forward	Ctrl + Shift + Enter	Session: Show next line in input history Edit: Reapply last change Trace: Skip current line
HLP *	Help	F1	Display the documentation for the system command, system name, control structure keyword or primitive glyph immediately to the left of the cursor
HO	Home Cursor	Linux: Ctrl + Home Mac: ⌘ + Up Arrow <i>or</i> ⌘ + Fn + Left Arrow Win: Ctrl + Home	Session: Move cursor to top left corner of Session log Edit: Move cursor to top left corner of content
IN	Insert Mode	Insert	Toggle between insert and overwrite modes
LBR *	Toggle Language Bar		Toggle display of the language bar (see section 5.3.2) at the top of the Session window
LC	Left Cursor	Left Arrow	Move cursor left one character
LL	Left Limit	Linux: Home Mac: Fn + Left Arrow Win: Home	Move cursor left as far as possible in current row
LN *	Toggle Line Numbers		Turn line numbers on/off in all windows of the same type as the active window.
NEW *	New Session	Ctrl + N	Starts a new Dyalog Session (a new instance of the interpreter)
PRF *	Show Preferences		Display the Preferences dialog box
PT	Paste	Linux: Ctrl + V Mac: ⌘ + V Win: Ctrl + V	Session: Paste the text contents of the clipboard at cursor Edit: Paste the text contents of the clipboard at cursor

QIT *	Quit Session	Ctrl + Q	Terminate the Dyalog Session
QT *	Quit	Shift + Escape	Session: Undo changes to a previously-entered expression that has not been re-executed and advance the cursor to the next line Edit: Close without saving changes
RC	Right Cursor	Right Arrow	Move cursor right one character
RD *	Redraw Function		Edit: Formats function to have correct indentation
RL	Right Limit	Linux: End Mac: Fn + Right Arrow Win: End	Move cursor right as far as possible in current row
RP *	Replace String	Ctrl + G	Edit: Replace. To do this, enter <RP> and type the string to replace the current search string with (see <SC>); enter <ER> to make the change. Enter <EP> to clear the field.
SC *	Search	Ctrl + F	Edit/Trace: Search. To do this, enter <SC>, type the string to search for and then enter <ER> to find the first occurrence of the string. Enter <EP> to clear the field. Also see related <NX>, <PV>, <RA>, <RP> and <RT>.
SI *	Strong Interrupt		Suspend code execution as soon as possible (generally after completing execution of the primitive currently being processed)
TB *	Tab Window	Ctrl + Tab	Move to next window in loop
TC *	Trace	Ctrl + Enter	Session: If there is a suspended function on the stack, open a Trace window for that function (<i>Naked Trace</i>)
TL *	Toggle Localisation	Ctrl + Up Arrow	Edit: For tradfns, the name under the cursor is added to or removed from the list of localised names on the function's header line
TO *	Toggle Outline		Edit: Open/Close outlined blocks (by default, outlines are shown)
UC	Up Cursor	Up Arrow	Move cursor up one character

UL	Up Limit	Linux: Ctrl + Home Mac: ⌘ + Up Arrow or ⌘ + Fn + Left Arrow Win: Ctrl + Home	Session: Move cursor to top left corner of Session log Edit: Move cursor to top left corner of content
US	Up Screen	Linux: Page Up Mac: Fn + Up Arrow Win: Page Up	Move cursor up one screen
WI *	Weak Interrupt		Suspend code execution at the start of the next line (generally after completing execution of the statement currently being processed)
ZMI *	Increase Font Size	Ctrl + =	Increase the size of the font in all the windows
ZMO *	Decrease Font Size	Ctrl + -	Decrease the size of the font in all the windows
ZMR *	Reset Font Size	Ctrl + 0	Reset the size of the font in all the windows to its default value.

Table 11. Dyalog keyboard shortcuts not currently supported on the RIDE

Keycode	Command	Description
BH	Run To Exit	Trace: Continue execution but show Trace window or stop on next error
CB	Clear Breakpoints	Edit: Clears all breakpoints from the current object Trace: Clears all breakpoints from the current object
FX	Fix	Edit: Fixes the function without closing the Edit window
HK	Hot Key (⌘SM)	⌘SM: With non-empty ⌘SM, toggle between ⌘SM window and Trace/Edit/Session window
JP	Jump	Toggle between Session window and current window
LS	Left Screen	Move cursor left one screen
MO	Move To Outline	Edit: When on the first or last line of a control structure, move to the opposite end (by default, outlines are not shown)

MR	Move/Resize	<p>Move and/or resize the current window. To do this, enter <MR> then:</p> <ul style="list-style-type: none"> • <LC>/<RC>/<UC>/<DC>: move window in specified direction • <LS>/<RS>/<US>/<DS>: move bottom right corner of window in selected direction relative to top left corner • <EP>: exit move/resize mode
MV	Move Block	Session: Move highlighted block of text to below the current line
NX	Next	Edit/Trace: When performing a Search/Replace, locate first match after current one
OP	Open Line	Edit: Opens a line underneath the current line
PV	Previous	Edit/Trace: When performing a Search/Replace, locate first match before current one
RA	Repeat All	Edit/Trace: When performing a Search/Replace, perform the same action again in all directions (Caution: replaces all matches in current object)
RM	Resume Execution	Trace: Continue execution and do not show Trace window or stop on next error
RS	Right Screen	Move cursor right one screen
RT	Repeat (Do)	Edit/Trace: When performing a Search/Replace, perform the same action again
SR	Screen Refresh	Redraw the session, removing all extraneous text that has come from external sources and resetting the session display
TG	Tag	Session: Tag (highlight) blocks of text. To do this, position the cursor at the start of the text to tag, enter <TG>, move to the end of the text to tag and enter <TG> again. Enter <TG> a third time to clear the current tagging.
ZM	Zoom	Toggle current window between current size and full screen size